LBNL-45863

Implementation and Testing of a Fault Detection Software Tool for Improving Control System Performance in a Large Commercial Building

T. I. Salsbury, Johnson Controls Inc.
R. C. Diamond, Lawrence Berkeley National Laboratory

ABSTRACT

This paper describes a model-based, feedforward control scheme that can detect faults in the controlled process and improve control performance over traditional PID control. The tool uses static simulation models of the system under control to generate feed-forward control action, which acts as a reference of correct operation. Faults that occur in the system cause discrepancies between the feedforward models and the controlled process. The scheme facilitates detection of faults by monitoring the level of these discrepancies. We present results from the first phase of tests on a dual-duct air-handling unit installed in a large office building in San Francisco. We demonstrate the ability of the tool to detect a number of pre-existing faults in the system and discuss practical issues related to implementation.

Introduction

Heating, ventilating, and air-conditioning (HVAC) systems are typically controlled using proportional plus integral (and sometimes plus derivative) PI(D) control law. In practice, HVAC systems exhibit non-linear operating characteristics, which cause control performance to vary when operating conditions change. Poor control performance can lead to occupant discomfort in the treated building, greater energy consumption, and increased wear on controlled elements, such as actuators, valves, and dampers.

In a conventional PI(D) feedback loop, the controller does not contain much information about the process it is controlling. Faults that lead to performance deterioration, or a change in system behavior, are often masked within a feedback loop. The control scheme described in this paper uses a model of the correctly operating system to supplement a conventional PI(D) feedback loop. The model is part of a feedforward control regime and acts as a reference of correct behavior, which facilitates the detection of faults that develop in the controlled system. Incorporation of a system model in the feedforward control scheme facilitates more consistent control performance as operating conditions change.

Several researchers (e.g. Gertler, 1998; Glass et al., 1994; Isermann, 1995; Patton et al., 1995) have proposed fault detection and diagnosis schemes based on the use of models. The main trade-off with model-based schemes is configuration effort versus model accuracy. Generally, the greater the potential accuracy of the models, the greater the effort required to configure the models for operation. In the proposed control scheme, we selected models that are configurable from design and commissioning information in order to reduce configuration effort. Previous work has shown that despite resultant loss of accuracy through model simplification, the scheme is capable of detecting a number of important faults and of improving control performance (Salsbury, 1999).

The Control and Fault Detection Scheme

Figure 1 shows the control and fault detection scheme. A conventional PI(D) feedback loop generates control action (u_{PI}) based on the error between the setpoint and the controlled variable. This feedback control action is then supplemented by a control signal (u_{FF}) generated by a simulation model(s), which is an *inverse* representation of the system. An inverse model predicts the inputs to a system based on measured outputs. The model is in static form and produces a control action appropriate for the current setpoint and measured disturbances. The control scheme is similar to one proposed by Hepworth and Dexter (1994), who used an adaptive neural network as the inverse system model.

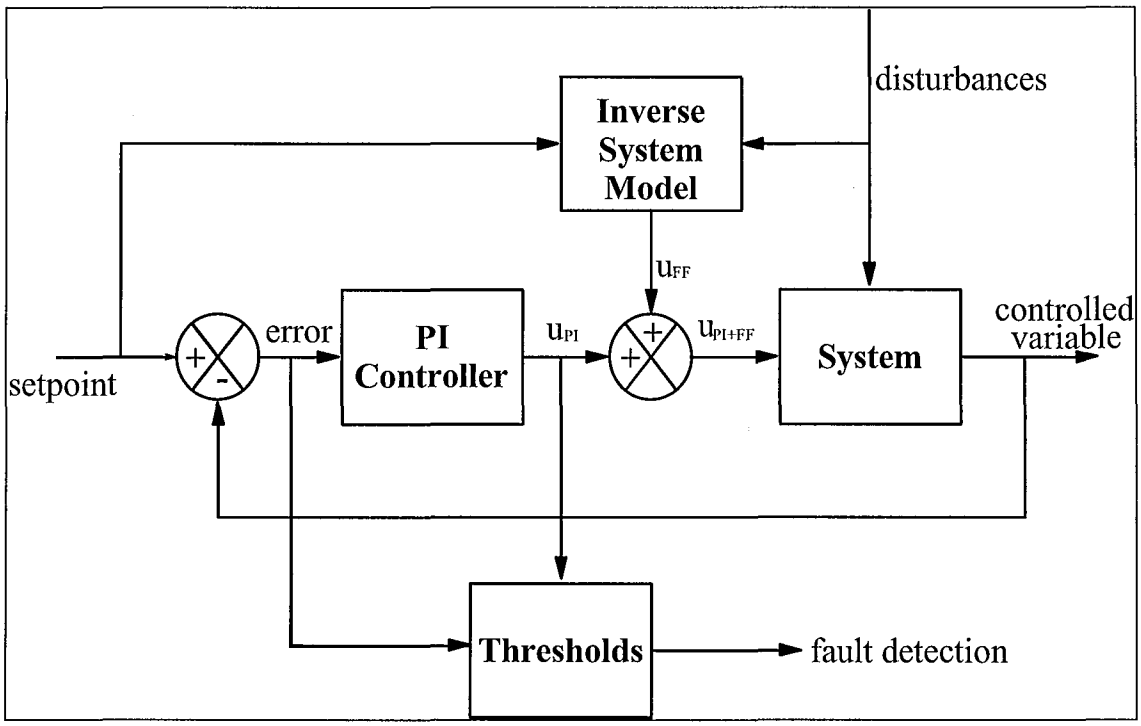

Figure 1. The Control and Fault Detection Scheme

The inverse model acting in isolation of the feedback loop would produce responses according to the open-loop dynamics of the system. Note that because the model is steady-state, the predicted control signal, u_{ff}, will change instantaneously for a change in any of the measured inputs. The feedback loop serves to speed the response time of the controller and eliminate offsets resulting from model inaccuracies and unmeasured disturbances. Assuming the effect of *unmeasured* disturbances is small, the (steady-state) feedback control action (u_{PI}) serves as an indication of the model/system mismatch. The control action, u_{PI}, thus represents an implicit measure of the difference between the predicted and the actual control signals for a particular setpoint. By configuring the model to represent a correctly operating system, the level of u_{PI} acts as an indication of fault development. Faults occurring in the system, which change its behavior or performance, thus create a mismatch between the model and system, leading to an increase in feedback control action.

The control scheme incorporates fault detection capabilities by monitoring the magnitudes of two indicator variables. The first indictor variable is the output from the PI controller (u_{PI}) - the "control signal error" and the second is the difference between the setpoint and the controlled variable – the "setpoint error". The controller generates an alarm if either of these variables exceeds a threshold for a sustained period.

The control signal error reveals changes caused by faults that do not affect the ability of the controller to maintain the setpoint, e.g., leakage through a control valve. A prolonged setpoint error that is not accompanied by a control signal error indicates a problem at or near to the point where the control signal would normally saturate, e.g., a capacity problem when full load is demanded. Simultaneous control signal and setpoint errors over a sustained period can indicate poor tuning or problems with the control loop. However, if the control loop is oscillatory, the errors may periodically return below their respective thresholds within a short enough time thereby avoiding alarm generation. Sensor errors are also detectable by the control scheme. Those that do not affect the ability of the control scheme to achieve the setpoint will be detectable through the control signal error. Large errors in the controlled variable sensor that cause the setpoint to become unattainable would also be detectable through the setpoint error.

The proposed control scheme triggers an alarm if the control signal error or setpoint error continuously exceed a threshold for a predetermined period. Figure 2 shows the fault detection algorithm. T_u is the threshold for the control signal error, T_e is the threshold for the setpoint error, and P is the maximum transgression period before generating an alarm. The fault detection part of the control scheme thus requires three parameters to configure it for operation: T_u, T_e, and P.

```
IF |u_PI|>T_u OR |error|>T_e,
        P=P+Δt
ELSE
        P=0
ENDIF
IF P>P_max
        FAULT=1
ELSE
        FAULT=0
ENDIF
```

Figure 2. Fault Detection Logic

Under a PI control regime, the setpoint error is supposed to reach zero in steady-state. T_e, can thus be selected heuristically based only on considerations of typical sensor noise and tolerable tracking errors. The parameter, P_{max}, relates to the maximum time between periods of steady-state. For HVAC applications, it is reasonable to assume that transience does not normally persist for more than 30 minutes between periods of (quasi) steady-state. We thus selected a value of 30 minutes for P. Selection of the threshold T_u is more difficult and relates to the accuracy of the models and the degree of detection sensitivity required. Ideally, T_u should be established through tests on the correctly operating system. However, as is

shown later, T_u may be also be set heuristically for preliminary testing in order to detect gross faults in the system.

Test System

Figure 3 depicts a schematic of the air-handling unit used in the tests, which is a dual-duct type having three thermal subsystems: mixing box, cooling coil, and heating coil. The air-handling unit has the capacity to deliver 74kg/s of air and provide 850kW of heating and 1260kW of cooling. The unit is installed in a large federal office building in San Francisco.

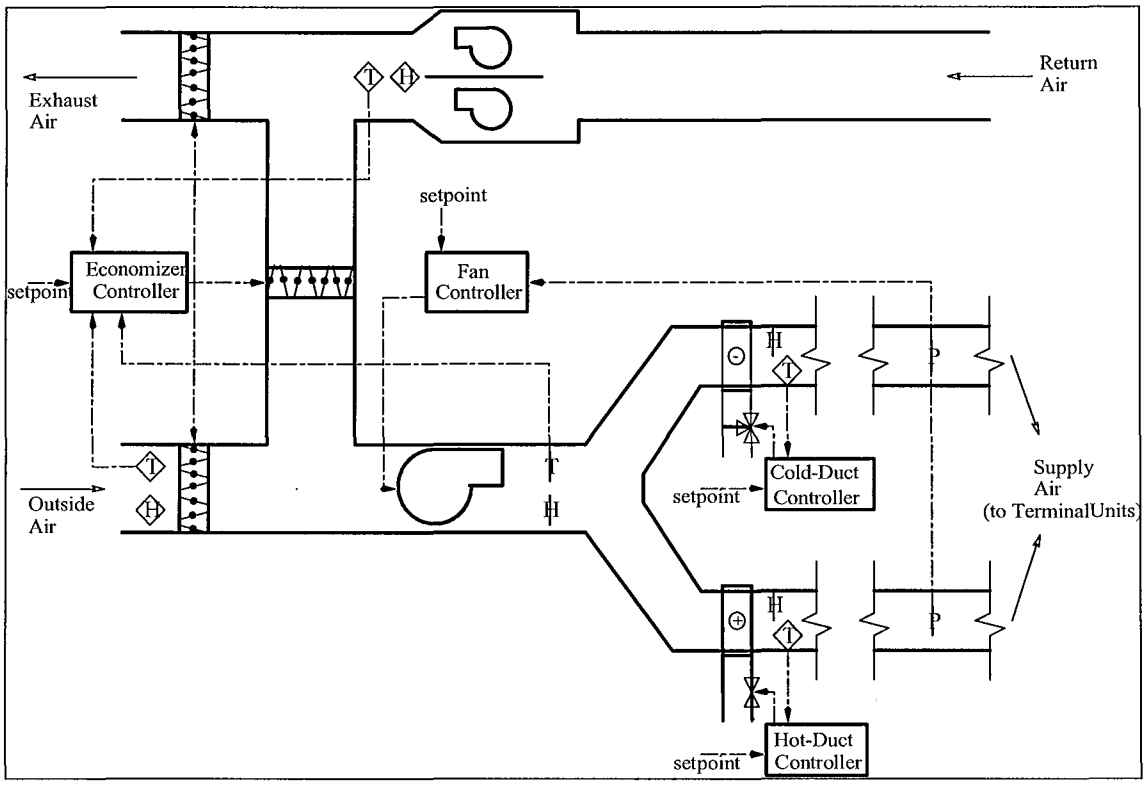

Figure 3. Schematic of the Dual-Duct Air-Handling Unit

Each thermal subsystem has its own controller. The mixing box controller modulates three sets of dampers in sequence to maintain mixed air conditions. There is a minimum outside-air requirement based on damper position (20% minimum outside-air) and a temperature-based economizer. The hot duct houses a steam-to-air heating coil regulated by a two-way valve, and there is a water-to-air cooling coil having a three-way valve in the cold duct. The fan speed varies according to load changes in the zones in a conventional VAV arrangement to maintain a constant static pressure in the supply ducts.

Simulation Models Used in the Control Scheme

The controller incorporates three separate models; one in each of the three separate control-loops in the air-handler: mixing box, heating coil, and cooling coil. Details of the

model equations can be found in (Salsbury, 1998). We simplified the models used in the feedforward controller in several respects. In particular, the models do not treat:

- Variations in coil thermal conductance with fluid flow rates;
- Dehumidification in the cooling process;
- Valve/damper non-linearity.

We make the latter simplification because characterization of this non-linearity requires parameters that are not easily obtainable or reliable, such as the inherent and installed characteristics of the valves and dampers. The simplification is reasonable, as one of the goals of the design and commissioning processes is to linearize the relationship between the control signal and controlled variable, e.g., by canceling coil non-linearity with valve non-linearity. Although the model simplifications reduce potential accuracy and performance of the scheme, a major advantage is that the parameter values may be obtained from typically available information, rather than requiring calibration data and additional tuning effort.

Table 1. Configuration Parameters

PARAMETER/DESIGN SPECIFICATIONS	UNITS
HEATING/COOLING COIL	
Heat transfer rate	kW
Cold fluid inlet air temperature	°C
Cold fluid mass flow rate	kgs^{-1}
Hot fluid inlet temperature	°C
Hot fluid mass flow rate	kgs^{-1}
MIXING BOX	
Minimum fractional outside air flow	%

Table 1 lists the parameters required by the models in the feedforward controller and Table 2 lists the required sensor measurements/variables. Note that in the dual-duct air handling unit, air temperatures and flow rates are required before the coils in both the hot and cold ducts.

Table 2. Required Sensor Signals/Variables

SENSOR SIGNAL	UNITS
Return air temperature	°C
Outside air temperature	°C
Air flow rates (hot and cold ducts)	kgs^{-1}
Pre-coil air temperatures (mixed air)	°C
Setpoints (mixed, hot-air, cold-air)	°C

Implementation

We developed the control and diagnostics algorithms into a stand-alone software program for testing with the test unit described earlier. We initially deployed the tool in a passive mode with the intention of validating the models and establishing thresholds.

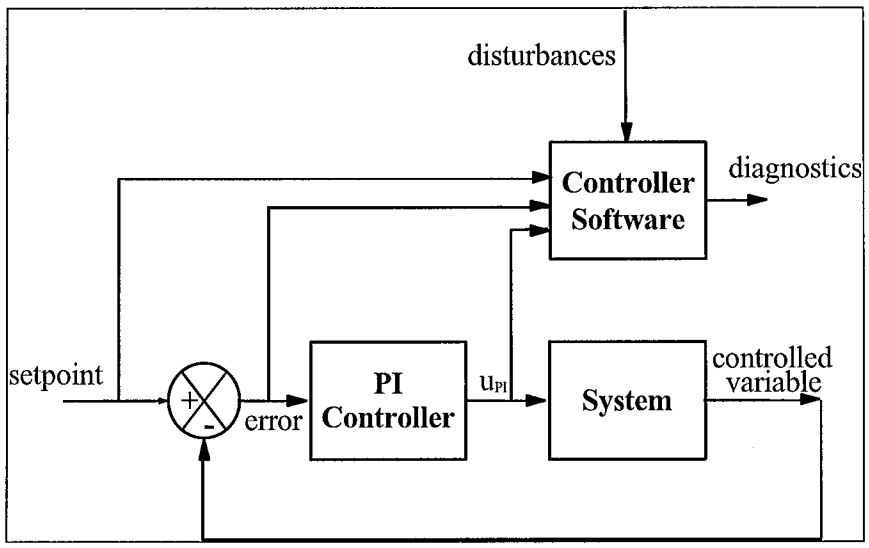

Figure 4. Interaction of Control Software with PI-Loop When in Passive Mode

Figure 4 depicts how the controller software was set up to interact with a PI loop in passive mode. In this mode, the feedforward control signals generated by the models do not affect the control operation and the system remains under PI-only control. In terms of fault detection, instead of using the PI control signal (u_{PI}) as a measure of the difference between the predicted and actual control signals, the difference is calculated explicitly, i.e., $u_{PI} - u_{FF}$.

Software Architecture and Connection to the EMCS

We developed the software based on three separate modules, as shown in **Figure 5**. The user interface provides diagnostic information to the user and allows the user to change parameters of the feedforward models, and other configuration information. The central module contains the control and diagnostics algorithms that function according to configuration information set by the user and data obtained from the energy management and control system (EMCS) network. The third module (control system interface) handles acquisition of data from the EMCS. The building in which we performed the tests was the subject of a recent large scale EMCS retrofit, which included replacing a large part of the system with BACnet (ASHRAE, 1995) compliant control devices. We thus developed the control system interface to use the BACnet communication protocol. Use of this communication protocol opens the way for testing the control software on any other BACnet compliant system regardless of the manufacturer.

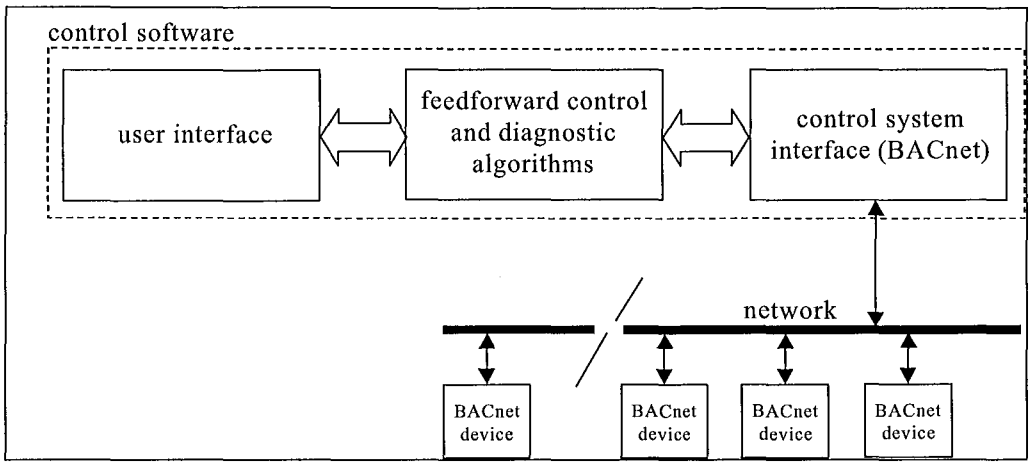

Figure 5. Software Module Interaction and Connection to the Control System

Figure 6 shows the user interface, which depicts the dual-duct air-handler used for the tests. Note that the two fans in the return duct have their speeds tracked to the speed of the supply fan, which is regulated in order to maintain the average of the hot- and cold-duct static pressures at a setpoint.

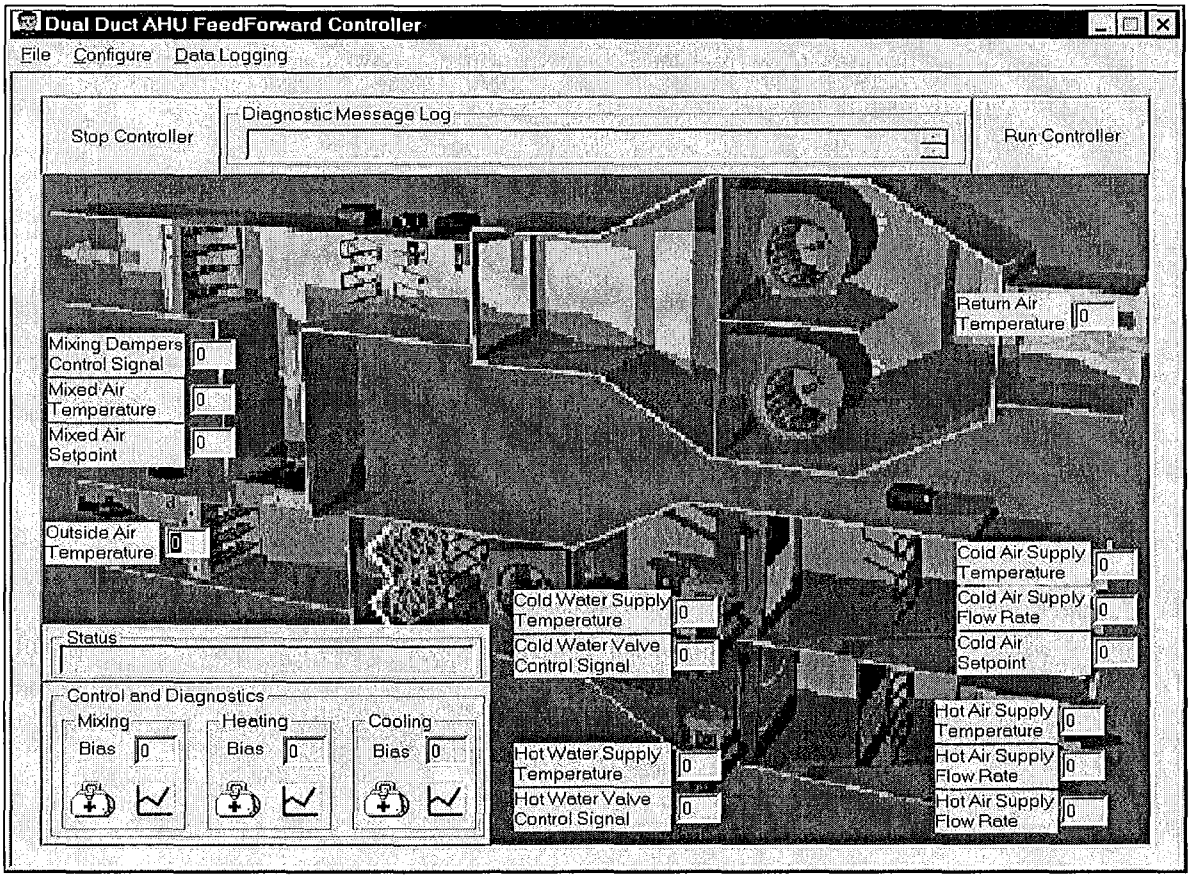

Figure 6. User Interface Showing the Dual-Duct Test Unit

Obtaining a Points List

Before we could carry out the tests, we had to obtain a point list of the sensors and control signals required by the software tool. This task is unavoidable for any application that needs to poll data from an EMCS network. The devices on the network that relate to the physical sensor and control-signal measurements required by the application require identification so they can be mapped onto the variables in the application program. The process of acquiring the necessary information can be both time consuming and subject to human error.

Sensor Availability and Accuracy

One problem encountered during the testing of the feedforward controller concerned sensor availability. In the test system, direct measurements of airflow in the hot and cold ducts were not available. The feedforward models use these measurements to calculate temperature rises/drops across the coils and the model predictions are quite sensitive to these variables. We therefore had to proxy the air flow rates using other sensor measurements that were available. We applied mass balances and pressure-flow relationships in addition to simple models to calculate airflow from the supply fan VFD control signal and static pressure measurements in the hot and cold ducts. The proxy was difficult to assess for accuracy, as we were only able to obtain point measurements of actual airflow at sporadic operating points. In addition, assumptions made in the proxy calculations introduced uncertainty into the predictions of airflow. Uncertainties in any of the measurements affect the performance of the control scheme and its fault detection sensitivity.

Test Results

The original aim of the first phase of testing was to validate the models in order to be able to establish the threshold values, T_u. However, it became apparent in the early stages of testing that blindly using data from the system in its "normal operation" state to set thresholds was inappropriate. We found that normal operation did not necessarily mean "correct operation". The initial test described in this section therefore entailed detecting pre-existing faults in the system. We discovered that the tool was useful as a re-commissioning aid and could be used in this way by setting T_u heuristically before carrying out the tests.

As explained earlier, the software operated in passive mode and maintained its fault detection capability by calculating the difference between the feedforward control signal and the measured PI control signal explicitly. During the test, the supply fan-speed and static pressures remained relatively constant, which reduced the potential errors stemming from the airflow proxy. In addition, the return and ambient air temperatures did not vary significantly during the tests. The effect on the AHU behavior from variations in measured disturbances was therefore small during the test period. **Figure 7** shows the return and ambient air temperatures and the airflow rate proxy in the hot and cold ducts.

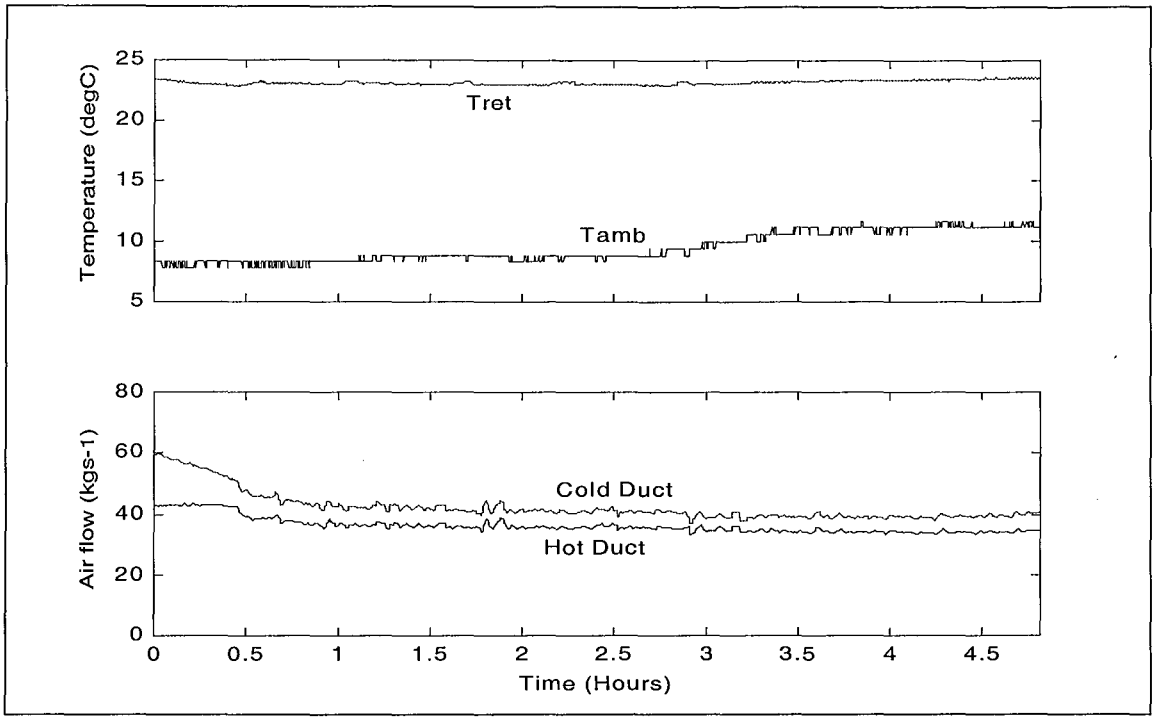

Figure 7. Measured Disturbances Affecting AHU Performance During Initial Test

Figure 8 and **Figure 9** show the test results. The top graph in **Figure 8** shows the controlled temperatures and their setpoints and the lower graph shows the control signals to each of the three subsystems. **Figure 9** shows the control signal errors and the setpoint tracking errors in the upper two graphs and the fault detector indicators in the three lower graphs. The first feature to note from **Figure 8** is that the controllers are unable to regulate at the setpoints very well, despite relatively constant measured disturbances and constant setpoints. The source of much of the instability appears to be the mixing box, which is cycling about its setpoint. This causes the mixed air temperature to vary, which in turn affects the load on the heating and cooling coils in their respective ducts downstream of the mixing process. The cooling coil reacts to the cycling in the mixing process with more extreme variations, causing the cooling valve to vary across its entire range. The disturbances in the mixing process influence the heating process to a lessor degree. However, the heating coil controller is still unable to regulate very well the controlled variable at the setpoint.

In **Figure 9**, operational problems in the AHU are evident with the indicator variables exceeding thresholds for sustained periods. Thresholds on the control signals and the controlled variables were set arbitrarily for this test and were thus not established empirically from training data. The control signal thresholds were set to 0.25 (25% of range) and the controlled variable thresholds to 2K. The cycling in the mixing process triggers an alarm due to the controlled variable being more than 2K outside of the setpoint for more than the half-hour time limit (P_{max} in **Figure 2**) set for the tests. Inspection of the mixing process revealed that leakage existed through the return-air dampers and this contributes to the control signal error exceeding the threshold at certain times, particularly when the controller demanded full outside-air ($u=1$).

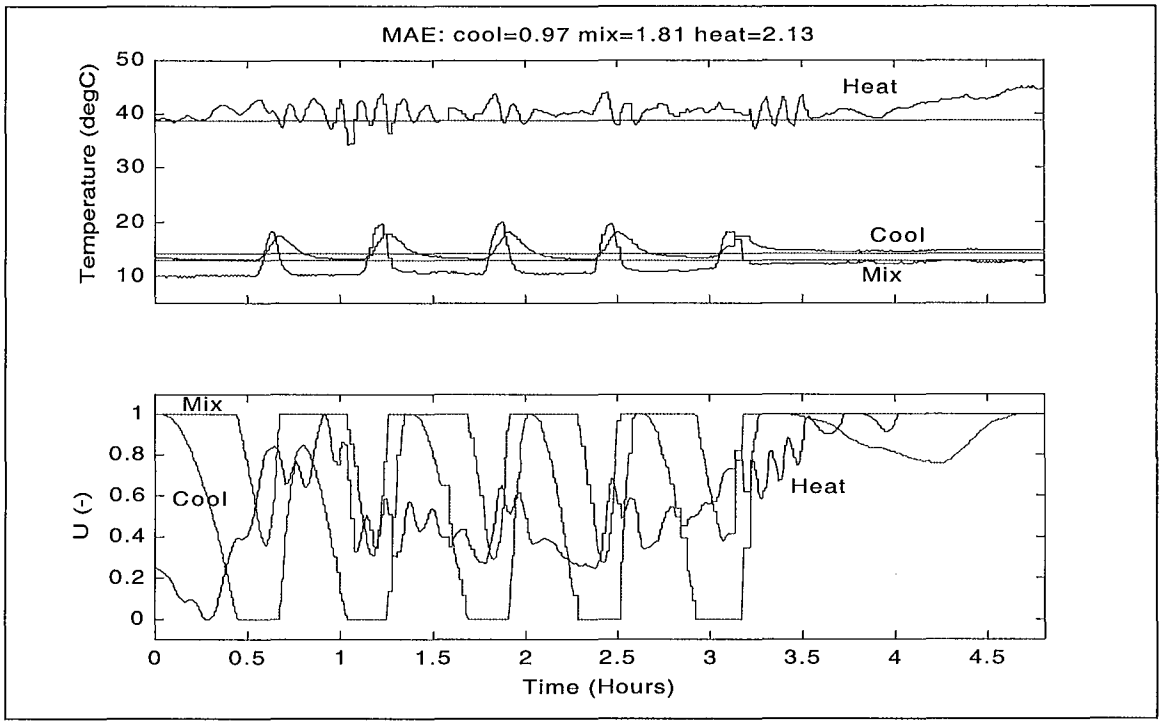

Figure 8. Control Performance

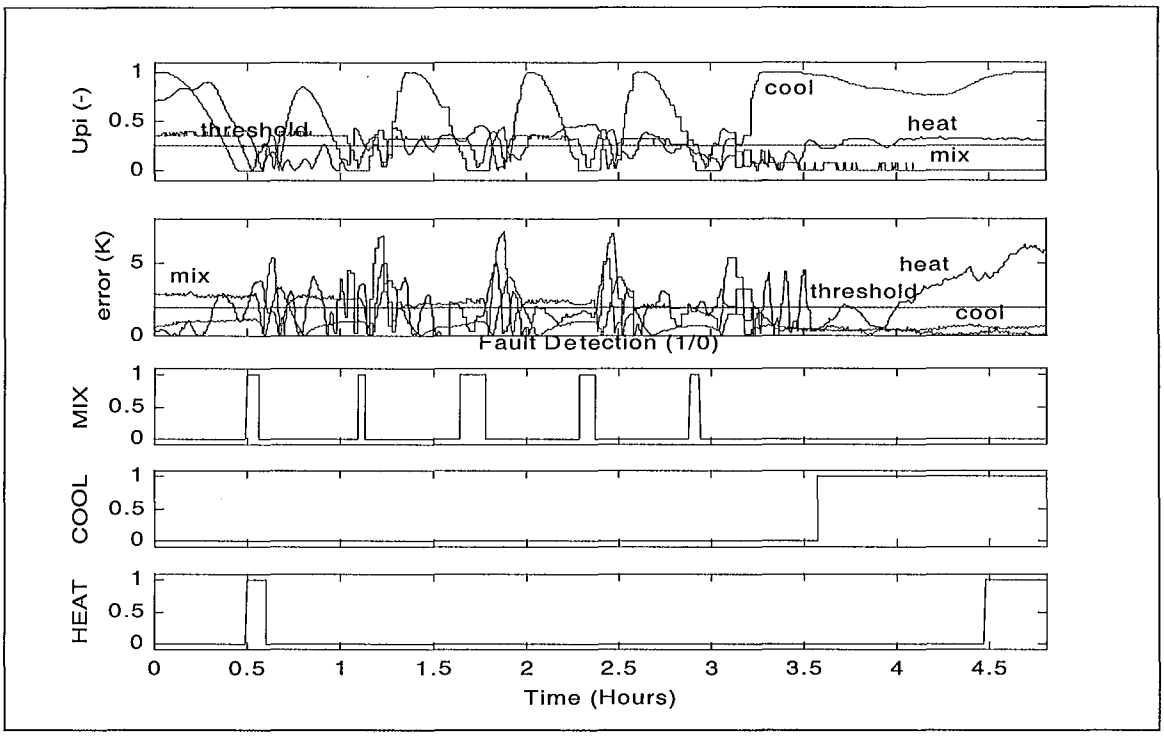

Figure 9. Fault Indicator Variables

Whenever the cooling process becomes active (i.e., the control signal is greater than zero), the error between the predicted and actual thresholds is large, as shown in **Figure 8**.

However, these large errors do not lead to an alarm, due to the cycling of the valve bringing the valve back to its closed position, where there is little prediction error before the half-hour time limit. The software thus only generates an alarm toward the end of the data when the coil valve stays open. The reason for the large error between the cooling control signal and the measured value was determined to be due to the chillers being disabled in the building during the test period. The cold water inlet temperature to the cooling coil was thus higher than expected as the cooling effect came only from the cooling towers. Since the cold-water temperature is a *parameter* in the controller software and not a variable input, the models predict a greater degree of cooling than is actually produced. The software therefore demonstrates a capability for detecting faults in the primary plant systems.

There are two periods in the test data when the software generates alarms for the heating coil system. The first alarm instance is caused by a sustained error between the predicted and actual control signals. Examination of **Figure 8** shows that in the period before the alarm, the heating valve is near or at its closed position while there is still a large difference in temperature across the coil. This behavior is inconsistent with the expectation of correct operation. The reason for the behavior is uncertain, but operators have reported leakage problems with the pneumatic valves controlling both the heating and cooling coils. The discrepancy between the predictions and measurements could thus be due to a large leakage through the valve. The second alarm instance is caused by simultaneous threshold transgressions in both the control signal and setpoint errors. The error between the controlled variable and the setpoint is quite significant as verified in **Figure 8**. It is possible that the simultaneous setpoint and control signal errors were due to a de-activation of the control loop, although we were unable to confirm this. The fact that the controller does not start to reduce the magnitude of the heating control signal as the setpoint error increases is strong evidence for a problem with the controller rather than the heating process.

Conclusions

This paper has described how simplified simulation models can be used to improve control performance and detect faults. Results from the first phase of tests on an AHU installed in a large office building demonstrated a fault detection capability and served to highlight practical implementation issues. We carried out an initial validation test of the controller software with the intention of establishing thresholds for later testing. However, we found that we could not reliably determine thresholds due to the existence of faults in the system. We therefore used the tool as a "re-commissioning" aid in order to detect the pre-existing faults, using thresholds selected heuristically before the tests. We detected the following problems in the test system:

- Poorly tuned economizer controller
- Leakage through the return air dampers
- De-activation of the chillers
- Valve leakage in the heating coil
- Heating coil controller deactivation/malfunction

The tests on the AHU demonstrated the difficulty in establishing a baseline of "correct operation" with which to determine thresholds and validate the models. A decision thus has to be made at the time of establishing thresholds whether to accept observed behavior as

being "correct" or to fix/tune the system to improve its performance. In the initial tests, the software proved useful as a re-commissioning tool allowing us to detect faults such as leaking valves and dampers. However, if these kind of faults were ignored by setting high threshold values the overall sensitivity of the tool would be reduced making new faults more difficult to detect. We therefore recommend that installation and tuning of the controller software take place following a thorough commissioning of the systems to ensure a fault-free starting condition.

Acknowledgements

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technology and Community Systems, and the Federal Energy Management Program, of the US Department of Energy under Contract No. DE-AC03-76SF00098.

References

ASHRAE, 1995. "Standard 135-1995 - BACnet - A Data Communication Protocol for Building Automation and Control Networks (ANSI approved)".

Clark, R. C. 1985. "HVACSIM+ Building Systems and Equipment Simulation Program Reference Manual". Published by the U.S. Department of Commerce, Gaithersburg, MD 20899.

Gertler, J. 1988. "Survey of Model-Based Failure Detection and Isolation in Complex Plants". IEEE Control Systems Magazine. Number 6. Volume 8. Page 3.

Glass, A. S., P. Gruber, M. Roos, and J. Tödtli. 1994. "Preliminary Evaluation of a Qualitative Model-Based Fault Detector for a Central Air-Handling Unit". Proceedings of 3rd IEEE Conference on Control Applications, Glasgow.

Hepworth, S. J. and A. L. Dexter. 1994. "Neural Network Control of a Non-Linear Heater Battery". Building Services Engineering Research and Technology. Volume 15. Number 3. Page119.

Isermann, R. 1995. "Model-Based Fault Detection and Diagnosis Methods". Proceedings of the American Control Conference, Seattle, Washington, USA. Page 1605.

Patton, R. J., J. Chen, S. B. Nielsen. 1995. "Model-Based Methods for Fault Diagnosis: Some Guidelines". Transactions of the Institute of Measurement and Control. Volume 17. Number 2. Page 73.

Salsbury, T. I. 1998. "A Temperature Controller for VAV Air-Handling Units Based on Simplified Physical Models". ASHRAE HVAC&R Research Journal. Volume 4. Number-3.

Salsbury, T. I. 1999. "A Controller for HVAC Systems with Embedded Fault Detection Capabilities Based on Simulation Models". Presented at the International Building Simulation Conference in Kyoto, Japan September, 1999.

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.