A+

Very good work.

STAT 248 – Final Project

Fault detection and Diagnostic for Rooftop Units

Abstract

Rooftop units fault detection and diagnostics is studied in this project. A rooftop unit is a packaged self-contained air conditioning unit which has the role of mixing the air and controlling the temperature and humidity of the air going to the building. It is the most common source of heating, cooling, and ventilation in small and medium commercial buildings, including retail stores, supermarkets, and restaurants. It is shown how the correlation among measuring parameters can be used to detect and isolate abnormalities in the system. The main challenge in rooftop unit diagnostics is to deal with measurement constraints coming from practical limitations. It is shown how such restrictions can be addressed systematically by intelligently using other measurements and filtering the effect of irrelevant components. The initial results seem to be promising in analyzing the rooftop unit performance without the need of detailed configuration data.

Good start

Introduction

Good purpose

Over the past two decades, there has been a growing interest on fault detection and diagnosis in Heating, Ventilation, and Air Conditioning (HVAC) systems. HVAC systems are a major consumer of energy in building. However, it has become apparent that only in a small percentage of facilities, HVAC systems are working efficiently or in accordance with the design intent [1]. The high proportion of inefficiently operating HVAC systems has led to both increased energy consumption and a large number of complaints by occupants. Studies have shown that operational faults in HVAC systems are one of the main reasons for inefficient performance of these systems [1].

A well known device in HVAC systems, Rooftop Unit (RTU), has the role of mixing the air and controlling the temperature and humidity of the air going to the building. A rooftop unit is a packaged self-contained air conditioning unit, typically mounted on the roof. It is the most common source of heating, cooling, and ventilation in small and medium commercial buildings, including retail stores, supermarkets, and restaurants. The U.S. Department of Energy estimates that rooftop and unitary air-conditioning equipment accounts for 62% of the annual energy consumption used for cooling of the current building stock of commercial buildings in the United States [1].

The focus of this project is on rooftop unit diagnostics. The questions of interest include:

do the relationships

I) How does the correlation among measuring parameters change when the system operates in a faulty mode?

II) Is there a logical pattern in parameter changes that can be used to isolate the fault source?

The data used in this project is from Target retail stores. Target and other retail chains are interested in automated diagnostics of the rooftop unit due to the extensive use of rooftop unit packages in their retail stores. In summer 2008, the performance of a few units located in Texas and California stores were gathered for diagnostic purposes.

A Brief Introduction to Rooftop Units (RTU)

Figure 1 shows the schematic diagram of a rooftop unit. Functionally, a rooftop air-conditioning unit (RTU) has the task of mixing and controlling the temperature and humidity of the air supplied to the building. It usually contains a fan, two dampers, a gas-fired heating system, and/or a direct expansion (DX) cooling system. The dampers control the mixing process of the return and outside air streams, and the heating and cooling systems control the temperature and humidity of the air going to the building.

The heating/cooling system may contain several stages: cooling stage 1, 2..., heating stage 1, 2... Each stage can be thought as an independent source of cooling or heating armed with an on-off controller. When it is on, certain amount of heating/ cooling energy is provided. The heating / cooling stages are activated by the rooftop unit controller sequentially. More information about the rooftop unit control system can be found at [2].

For convenience, the following abbreviations are used in this report:

RTU: rooftop unit

RAT: return air temperature, the temperature of the air coming out from the building

OAT: outside air temperature

DAT: discharge air temperature, the temperature of the air going to the building

MAT: mixed air temperature, the temperature of the mixed air between the outside and return air where is it in Figure 1 ?

CL1, CL2 …: cooling stage 1, 2 …

HT1, HT2 …: heating stage 1, 2 …

The main challenge in RTU diagnostics is to deal with constraints coming from practical limitations. In real applications, there are limited measurements available to monitor RTU performance. For instance, there is usually no reliable measurement for the mixed air temperature (MAT)[1], which forces us to rely on DAT sensor to analyze the performance of both dampers and cooling/ heating systems. In such a scenario, when the sensor output is contaminated, it could be due to the mal-functionality of either any involving components, and it may not be straightforward to isolate. Wording ?

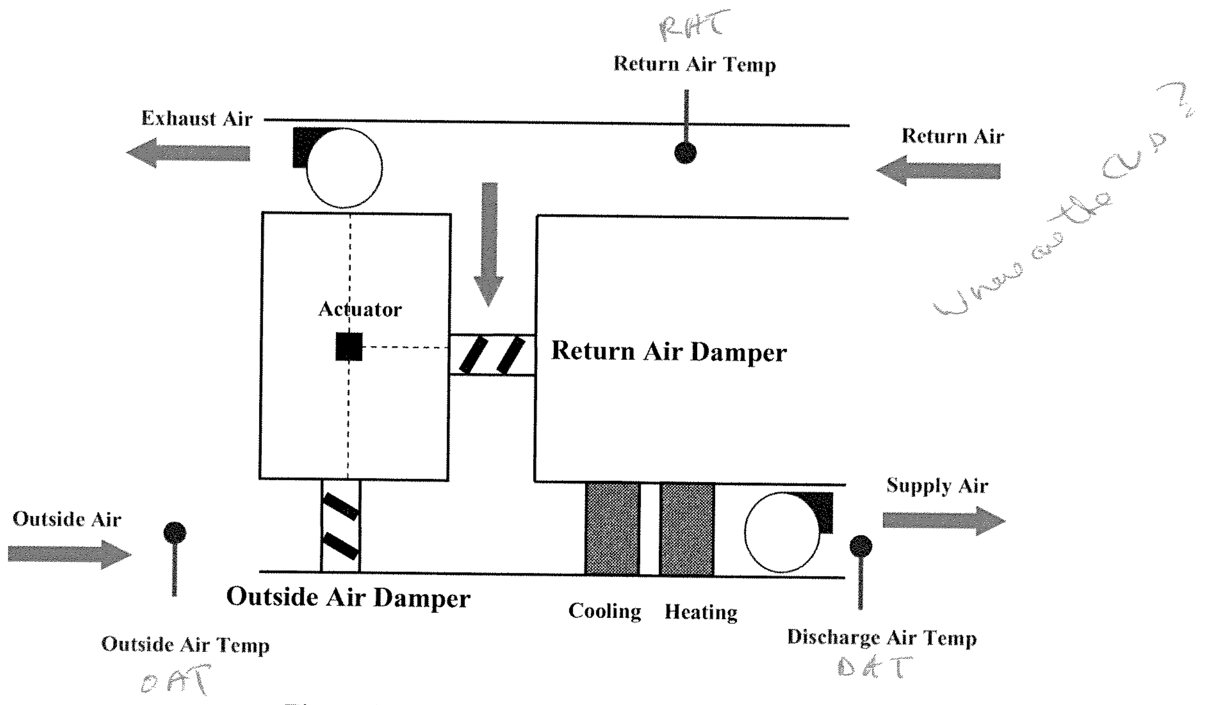


Figure 1. Rooftop unit schematic diagram

Add legend

[1] This is a well known issue in RTU operation. Due to practical limitations, usually the mixed air temperature sensors are mounted where the mixing process between the return and outside air are not complete yet. Therefore, the sensor readings have bias.

Statistical papers usually do not have footnotes.

The goal of this project is to develop diagnostic solutions within the boundaries defined by practical limitations. In other words, the aim is to make the best use of available measurements for diagnostic purposes.

Rooftop Unit Diagnostics

Figure 2 shows the performance of a rooftop unit from a Target store in Texas. The data contains 24 hr operation of the unit starting at 12:00 07/31/2008 with 1 minute sampling interval rate. The first graph shows the variations of OAT, DAT, and RAT while the second, third, and forth graphs show the variations of CL1, CL2, CL3, and damper (outside air damper[2]). The damper position has been divided by 100. So, when it shows the damper position at 0.3, it means the damper is 0.3*100 = 30% open. Also, as it was a hot summer day in Texas, naturally the heating system was off.

Discuss details of plots, esp. initial periodicity in DAT.

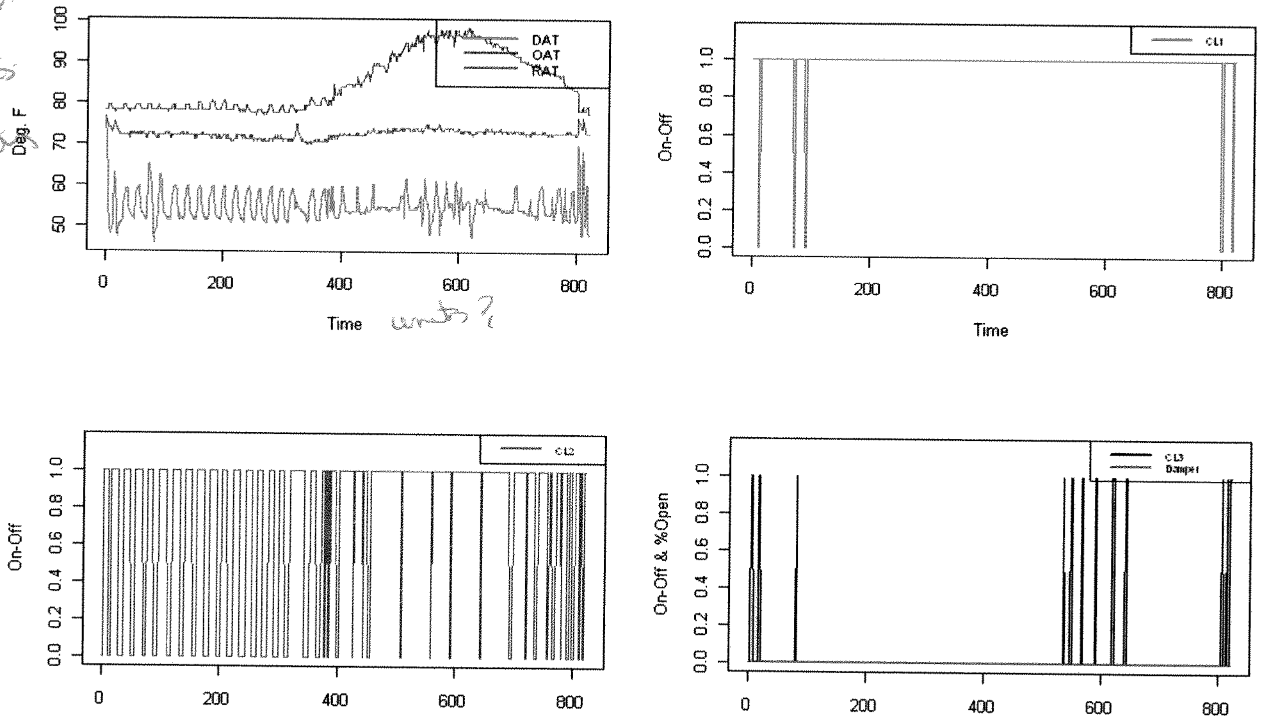


Figure 2. RTU performance from a Target store in Texas, from 07/31/08 0:00 to 07/31/2008 23:55, 1 minute sampling rate interval

Say what each panel is.

[2] The outside air and return air dampers are manipulated by the same actuator reversely; when one is opening, the other one is closing. For example, when the outside air damper is at 30% open, the return air damper is at 70% open.

Does this show up in the data?

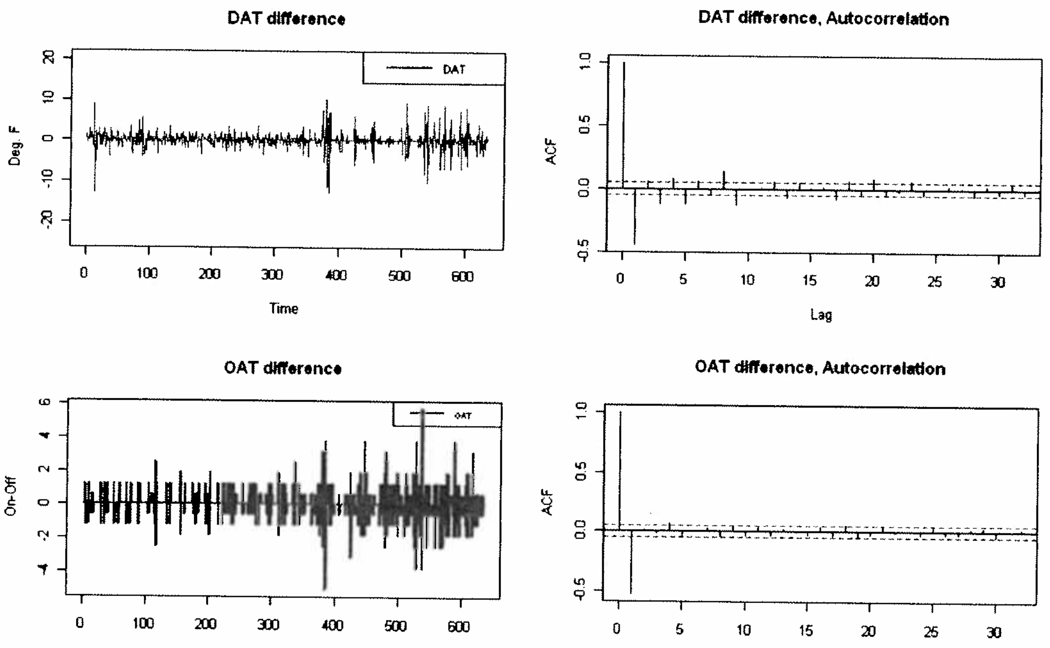


Figure 3. DAT and OAT and the autocorrelation functions after filtering

To make the data more stationary, a difference operator of order two was applied. Define As an example, Figure 3 shows the filtered version of DAT and OAT and the corresponding autocorrelations. You see that the filtered data is closer to a stationary process. For the rest of this report, to motivate calculations, it is assumed that the filtered data is stationary.

Now, as the first step, let's focus on the functionality of heating/ cooling systems. The question to ask is: how the mal-functionality of heating/ cooling systems can be detected through monitoring DAT variations? A very common fault in RTU operation is to have different stages of heating/ cooling systems broken[3], and the aim here is to detect such mal-functionality automatically.

The proposed approach is to analyze the linear correlation between the cooling/ heating command and DAT variations. This inherently assumes that the effect of heating/ cooling systems on DAT variations is linear. In reality, the relationship has some non-linear aspects as well, but we are hoping that such simplification is acceptable for diagnostic

Describe in words

[3] A broken heating / cooling stage means here that it does not turn on when it is commanded by the RTU controller.

purposes[4]. In order to incorporate all stages of cooling and heating, a set of new time series parameters $C_1, C_2, C_3, H_1, H_2, H_3$ are defined as:

$$C_1(t) = \begin{cases} +1 & if \quad CL_1(t) \quad is \quad On \\ 0 & if \quad CL_1(t) \quad is \quad Off \end{cases} \qquad H_1(t) = \begin{cases} -1 & if \quad HT(t)_1 \quad is \quad On \\ 0 & if \quad HT(t)_1 \quad is \quad Off \end{cases}$$

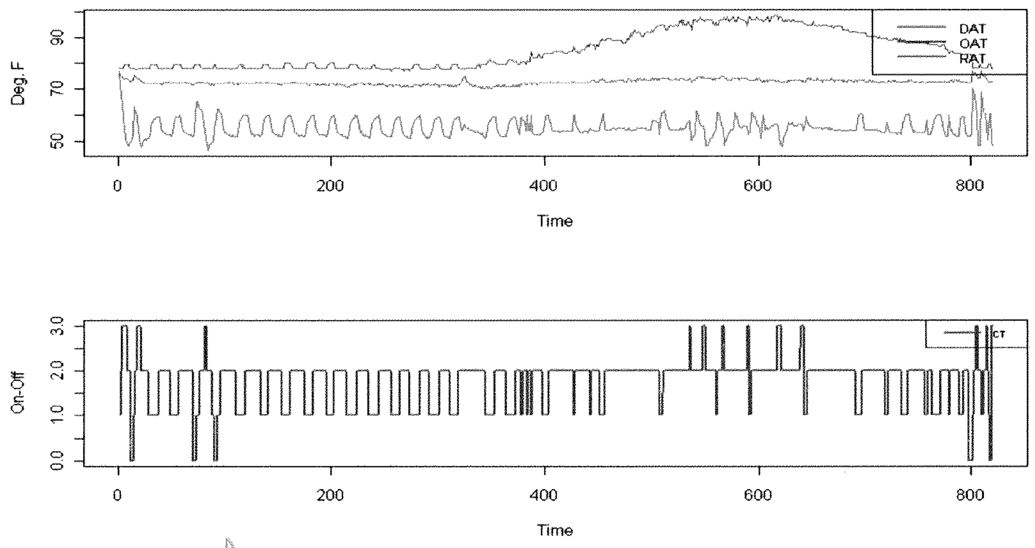
Same for C_2,C_3, H_2 and H_3. Now, CT is defined as:

why computed?

$$CT = C_1 + C_2 + C_3 + H_1 + H_2 + H_3 \qquad (1)$$

, which is the arithmetic summation of all heating/cooling stages. In Figure 4, you can see the calculated CT (before applying the difference operator) for the data shown in Figure 2.

Discuss



Needs a legend.

Figure 4

Now, the linear dependency between CT and DAT can be evaluated through the coherency and phase functions which are shown in the third and forth graphs of Figure 5.

---

[4] The basic equation of thermal interaction between a heating/cooling coil and the air (two objects with different temperature) can be expressed as [3]

$$q = hA(T_s - T_a)$$

$q, A, h, T_s, and T_a$ are heat transfer rate, surface area, heat transfer coefficient, coil surface temperature, and the temperature of the air respectively. This is a very simplified expression of thermal interaction process happening inside RTU. Now, if it is assumed that q, h, A, and Ts are constant (which may not be necessarily true all the time), you see that the relation between the air temperature (*Ta*) and heating/heating command (*q*) is linear.

Confusing. Make a table

Note how these two variables are highly correlated in low frequencies as expected: the frequency of on-off commands can not pass certain threshold as there is usually a minimum delay (changing from case to case) between two successive turn on commands. Also, note how the phase diagram shows a lead for CT over DAT in low frequencies. This is coming from the fact that when a cooling / heating stage is turned on, it will take a few minutes (3-5 minutes) until DAT fully respond to that because of warm up period. Estimate?

Relate to slope in Fig 5

The level of dependency is expected to decay when one or more cooling stages are broken. This will be shown through another example later A question that one might ask at this point is how the broken cooling/heating stage can be isolated. In other words, after the detection of a problem in the cooling/heating system, the next step is to locate the source of the problem (the broken stage). ✓

The proposed solution is to compare the correlation between DAT and a set of CT's, in which, each CT is developed based on the assumption that one or more cooling/heating stages are broken. When a cooling stage is assumed to be broken, the corresponding C in Equation 1 is set to zero regardless of the CL values. The CT set should contain all possible combinations of the cooling/ heating system faults. The CT with strongest correlation with DAT will be considered as the winner and the corresponding hypothesis will be concluded as the health status of the heating/ cooling system. Coherence?

Provide formula for clarity.

For instance, in Figure 7, the graphs on the left side (first, third, and fifth ones) show a set of CT's (before applying the difference operator) --- the first graph assumes there is not fault in the system, the second graph assumes CL2 is broken, and the third graph assume CL1 and CL2 are broken[5] --- and the graphs on the right side show the corresponding coherency. As you see shows the strongest correlation with DAT as expected.

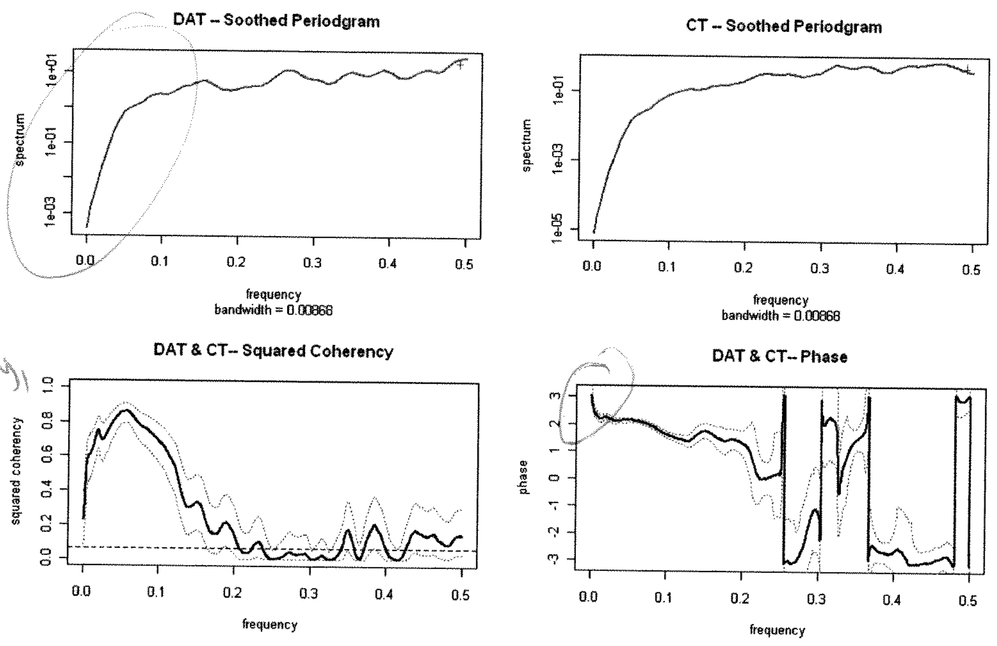
Is Figure 6 discussed? what was known about what's broken?

Let's test the developed diagnostic mechanism with another dataset. Figure 7 shows 24 hr performance of another rooftop unit this time located in a California store. Again the data was recorded in summer 2008. Just by looking at the graphs, you notice that DAT does not respond accordingly to cooling stage 1 (CL1) command. Be specific

Say why.

[5] Due to the space limitation, only few hypotheses are shown here.

Discuss



You have squared coherency, coherency and coherence - Be clean.

Discuss stat of it,

Figure 5. The power spectrum and coherency of the data shown in Figure 2

This gives the idea that CL1 might be broken. Figure 8 shows different CT's and the corresponding coherency graphs. As you see, the second graph, the one that corresponds to the assumption of broken CL1, clearly has the strongest correlation which confirms the idea of broken cooling stage 1.

After analyzing the functionality of cooling/ heating systems, the next step is to evaluate the performance of the dampers. In other words, the next question is how the mal-functionality of a mixing box[6] can be detected and isolated. The common faults here are: reverse damper (damper closes when it supposed to open), stuck (damper is stuck at a position), leakage (damper does not fully close), etc. ?

Could you design a diagram?

As explained earlier, the challenge here is the lack of a reliable measurement for the mixed air temperature (MAT). The only available measurement getting affected by the mixing box operation is DAT sensor. But DAT is also affected by the heating/cooling system which makes it difficult to discriminate between these two.

[6] Mixing box is a component of RTU in which the mixing process happens. It contains the outside air and return air dampers, and the space where mixing occurs.

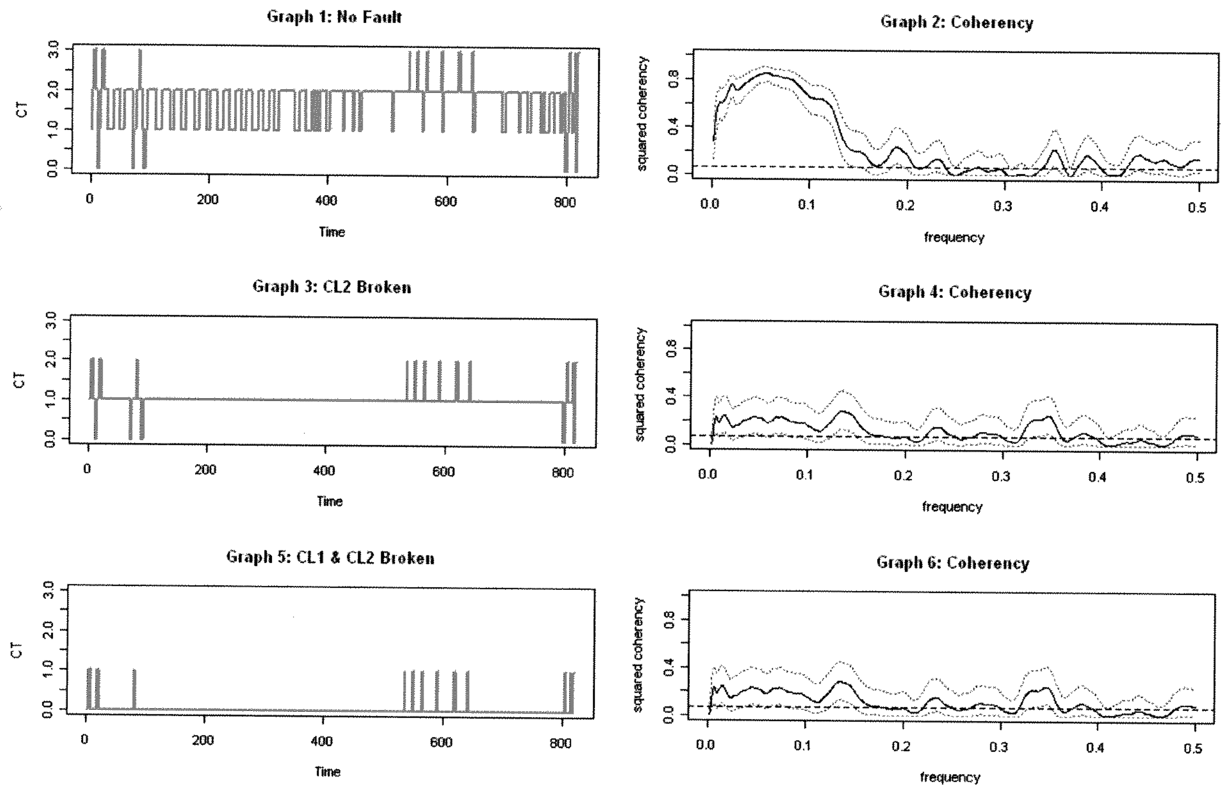


Figure 6.

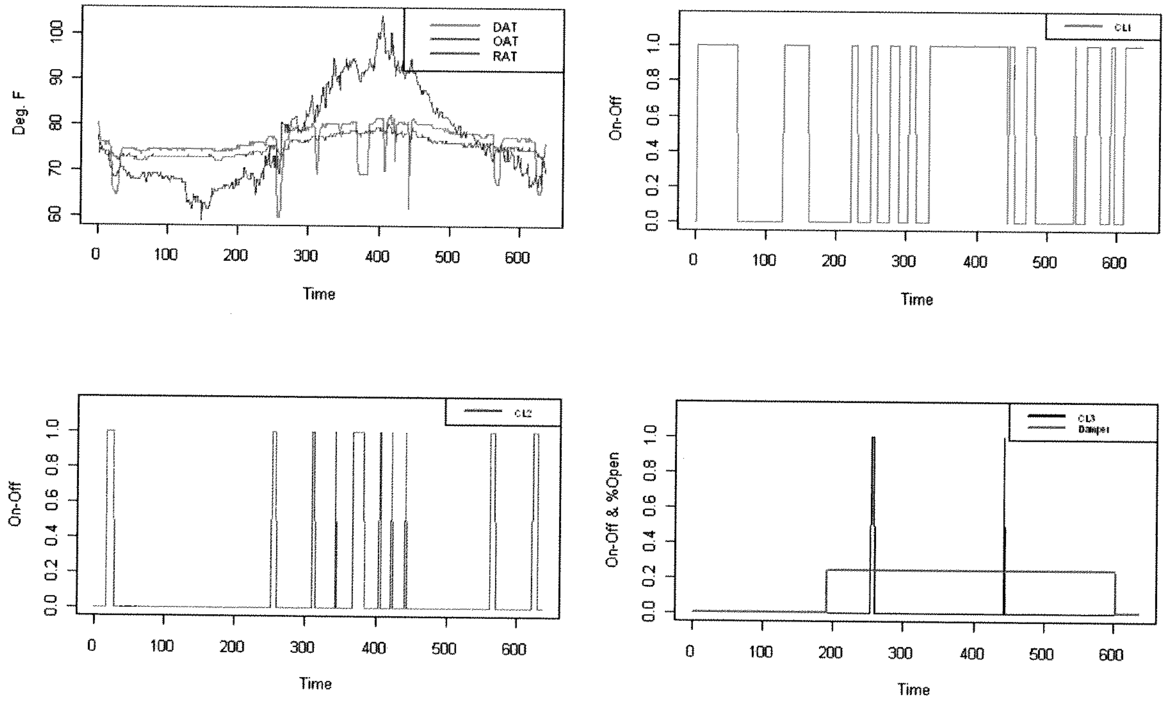


Figure 7. RTU performance from a Target store in California, from 08/15/08 0:32 to 08/15/2008 23:55, 1 minute sampling rate interval

You really need legends for your figures.

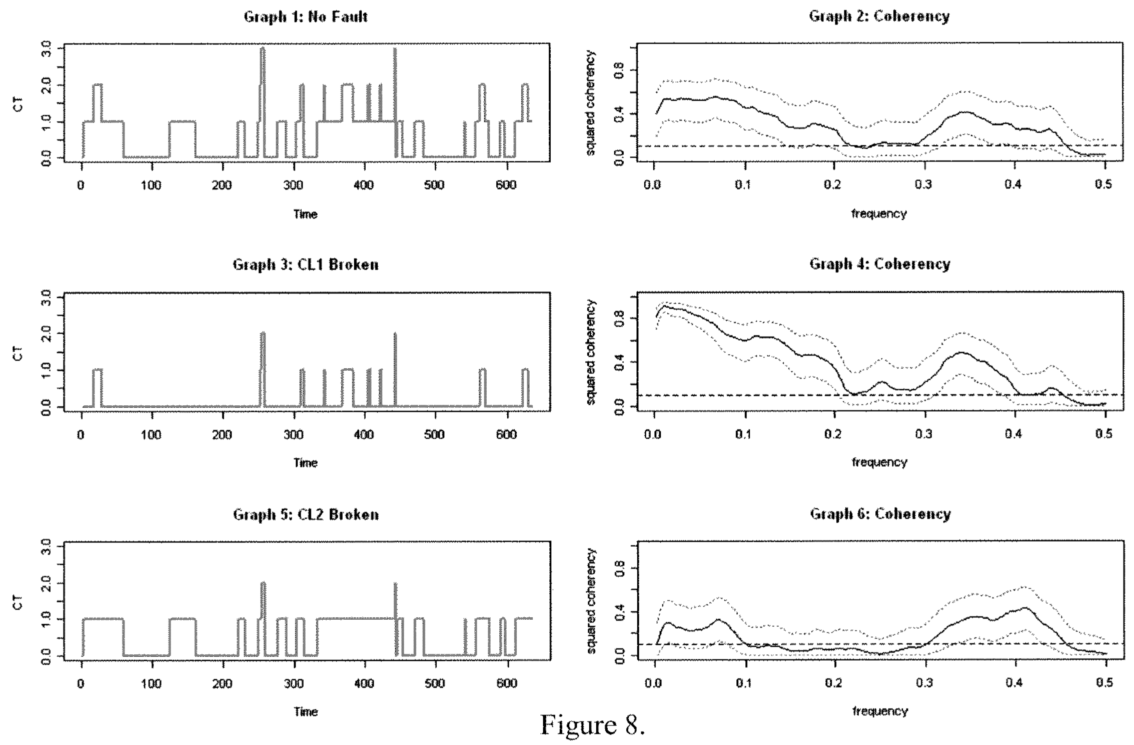


Figure 8.

One solution is to evaluate the performance of the mixing box when the heating / cooling system is off and has no effect on DAT. The problem this solution is that such a scenario may rarely happen --- especially if it is a hot or cold season. Another approach, the approach proposed here, is to take out the effect of the heating/ cooling system from DAT, and then use the filtered DAT to evaluate the mixing box performance. Say how.

What's this? With the assumption of linear correlation between the heating/cooling system and DAT, this can be achieved by estimating the function $\{\beta_t\}$ which minimizes the following equation:

$$MSE = E\left(DAT_t - \sum_{r=-\infty}^{\infty} \beta_t CT_{t-r}\right)^2 \quad (2)$$

? r

One-sided case?, Working in real time?

DAT$_t$ is the value of DAT at time t and CT was defined in equation 1. As it is shown in [5], the Fourier transform of $\{\beta_t\}$ can be found by:

$$\hat{\beta}(w_k) = \frac{\hat{f}_{DAT,CT}(w_k)}{\hat{f}_{CT,CT}(w_k)} \qquad (4)$$

, and then $\hat{\beta}_t$ can be estimated through the inverse Fourier transformation of $\hat{\beta}(w)$. The first graph of Figure 9 shows the estimated $\hat{\beta}_t$ for the data shown in Figure 2. A possible model for this system could be:

State of these coefficients, were found explicitly.

$$DAT_t = -0.93\times CT - 0.98\times CT_{t-1} - 2.09\times CT_{t-2} - 1.23\times CT_{t-3} - 0.95\times CT_{t-4} + 0.83\times CT_{t+1} \qquad (5)$$

, and it is consistent with the functionality of the cooling system functionality: you see that DAT not only depends on the current value of CT but past values as well (3-4 lags). As mentioned earlier, when a cooling stage is turned on, it normally takes three to five minutes (3-4 lag) to fully affect DAT. You also see that DAT depends on future values of CT in opposite direction, which is coming from the fact that when a cooling stage is turned off, DAT will be increasing for a few minutes as the cooling effect is diminishing.

Good thoughts. What's the causal connection?

Good thoughts.

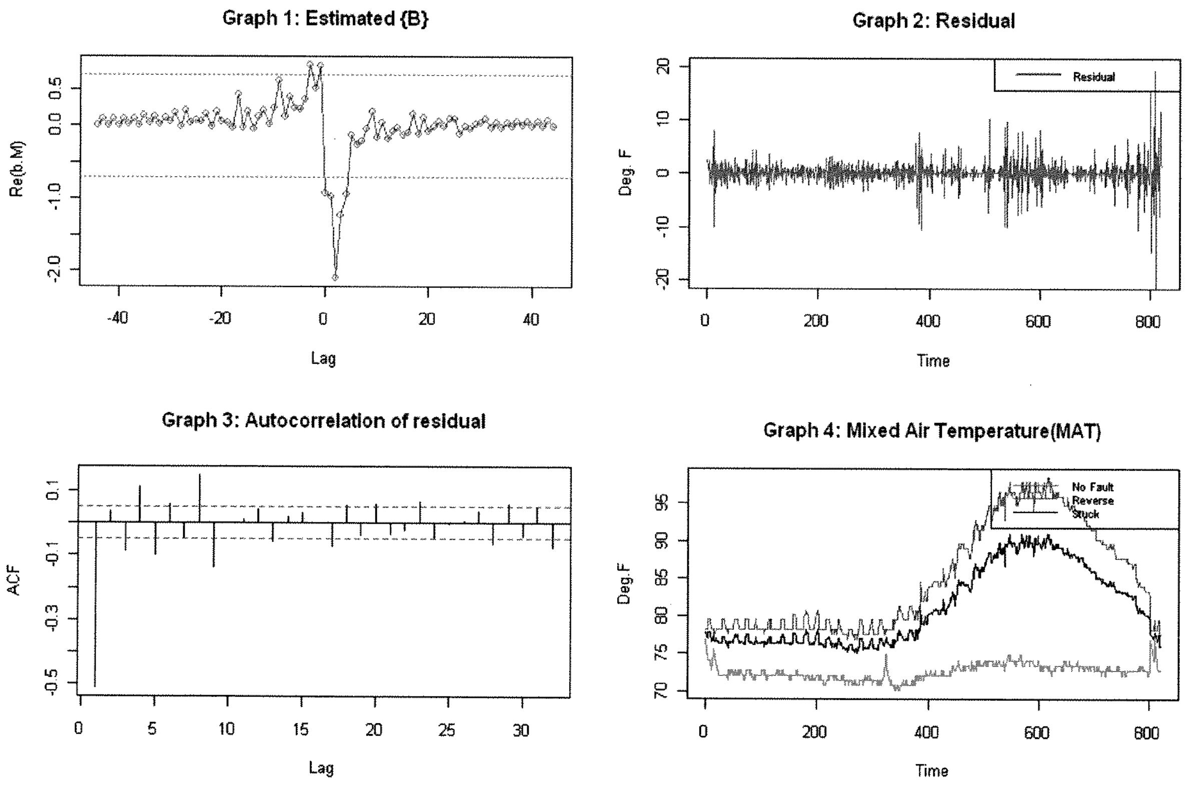


Figure 9.

The second and third graphs in Figure 9, show the residual, w_t, and autocorrelation of the residual:

$$w_t = DAT_t - \sum_{r=-\infty}^{\infty} \beta_t CH_{t-r}$$ ✓

Figure 10 shows similar results for the data shown in Figure 6. Based on the first graph of Figure 9, the possible model can be:

$$DAT_t = -3.12 \times CT - 2.3 \times CT_{t-1} - 1.43 \times CT_{t-2} - 1.1 \times CT_{t-3} + 1.29 \times CT_{t+1} + 0.70 \times CT_{t+2} + 0.86 \times CT_{t+4} \tag{6}$$

and again you can have the same interpretations as for Equation 5.

Looking at autocorrelation functions in figures 9 and 10, you notice that the residual is not a white noise. This should not be a surprise. The residual is expected to be a rough estimation of the mixed air temperature (MAT), and its characteristics depend on OAT, RAT, and the damper position.

Now, the question is how the estimated residual can be used to evaluate the functionality of the mixing box. Logically, the correlation among MAT and OAT/ RAT should change accordingly as the damper position changes. When this does not happen, it is an indication of fault. Note that although in Figures 2 and 6 dampers seem to not changing frequently, this is not necessarily true all the time. RTU controller may manipulate the damper as frequent as the cooling/ heating system.

Again, the proposed approach is to compare the correlation between the residual and a set of set of MAT's, in which each MAT is generated based on the assumption of one or more faults in damper operations. As before, the MAT that shows the strongest correlation with the residual (which is an estimation of the true MAT) will be chosen as the winner and the corresponding assumption will be considered as the dampers functionality.

A simplified model of the mixing box operation is:

Present an argument.

$$MAT = (DMP/100) * OAT + (1 - DMP/100) * RAT$$

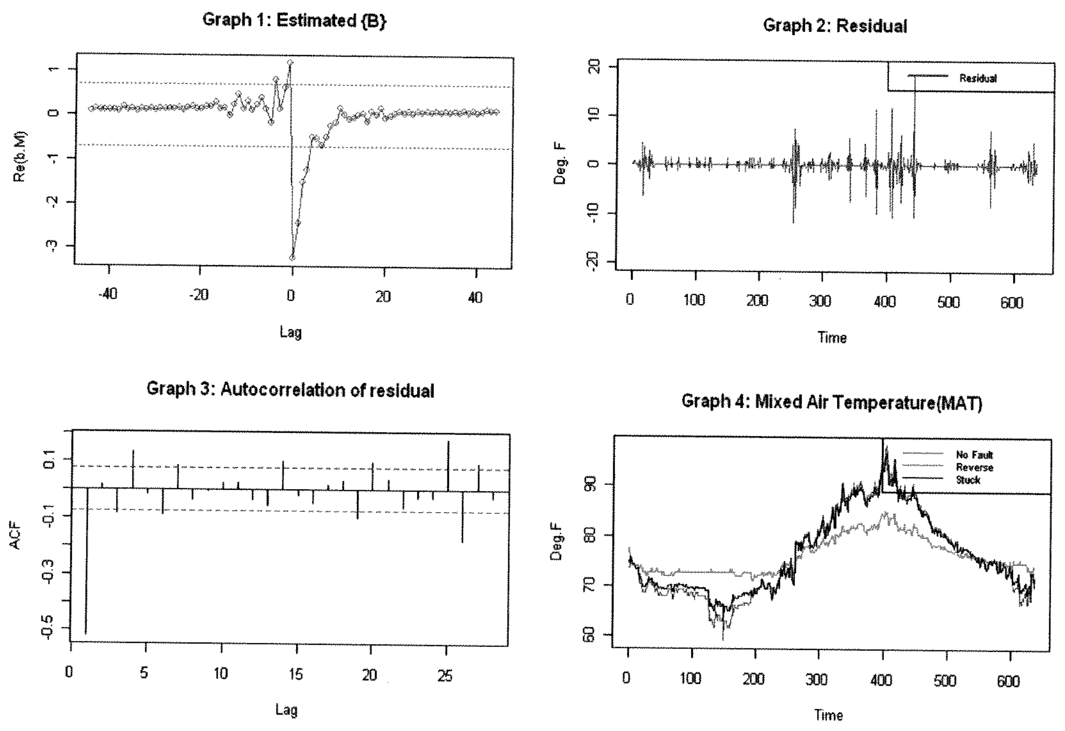


Figure 10.

A legend is essential here.

In reality, there are other factors involved but hard to measure. It is assumed that this level of simplicity is acceptable for diagnostic purposes. This equation can be manipulated to calculate MAT in different fault modes. The forth graph of Figures 9 and 10 shows the calculated MAT in different fault modes for the data of Figures 2 and 6 respectively.

Figure 11 shows the coherency and phase between the residual and calculated MAT's based on the data in Figure 2. As you see, the first row, which corresponds to no fault condition, shows the higher correlation and almost zero phase in low frequencies. Note that the variations of OAT, RAT, and DMP usually happen in low frequencies, and the high frequencies normally correspond to noise in the system. Figure 12 shows the cross-correlation function of the same pairs after prewhitening. Again you see that the case of no fault, first graph, has the highest correlation at lag zero which confirms the idea of no fault in dampers operation. Figure 13 shows the same analysis for the data in Figure 6.

Need to zoom

Are these bumps at pertinent frequencies?

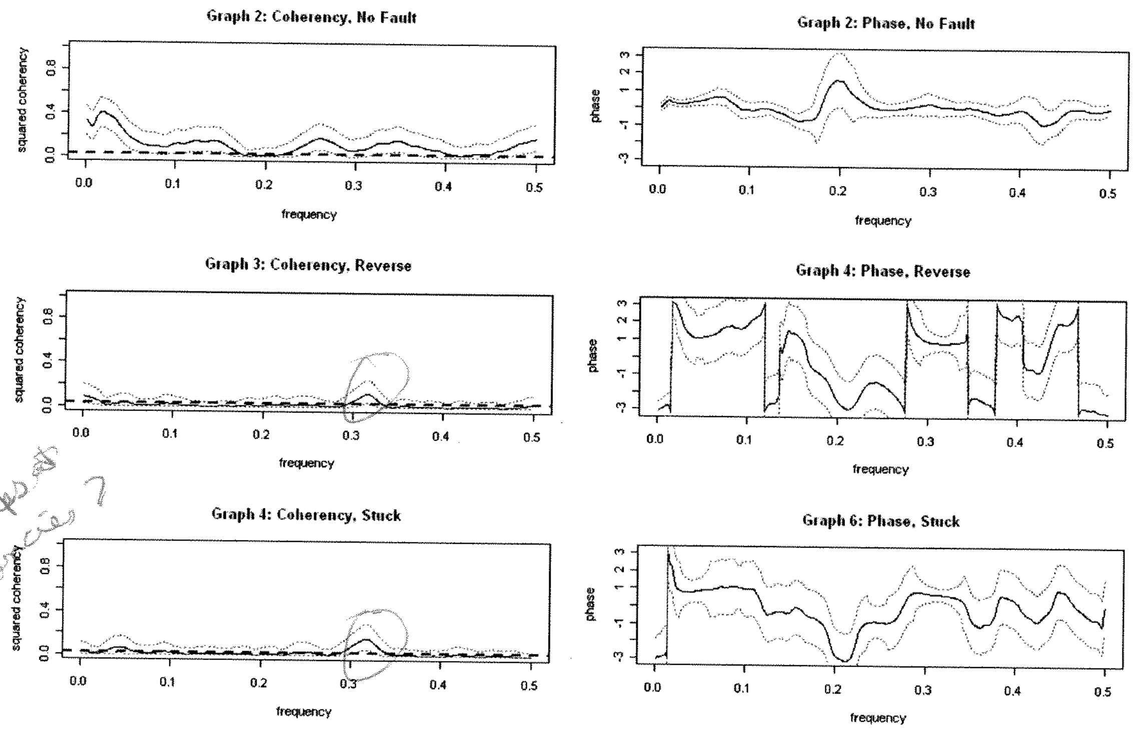


Figure 11. The coherency and phase between the residual and different estimations of MAT for data in Figure 2

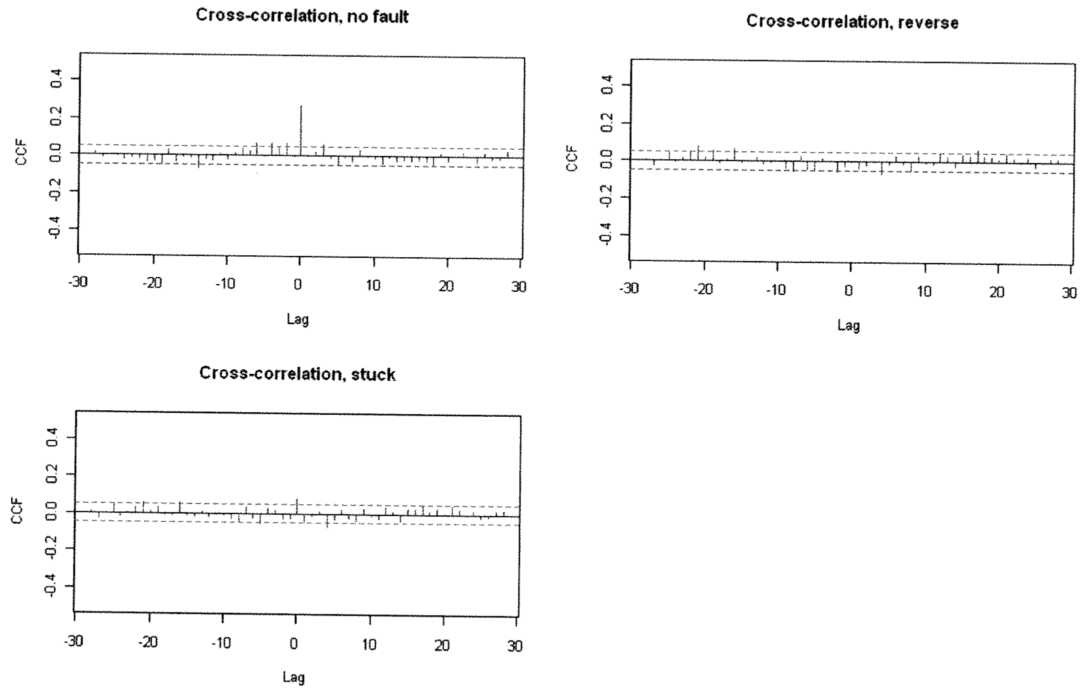


Figure 12. The cross-correlation between the residual and different estimations of MAT for data in Figure 2

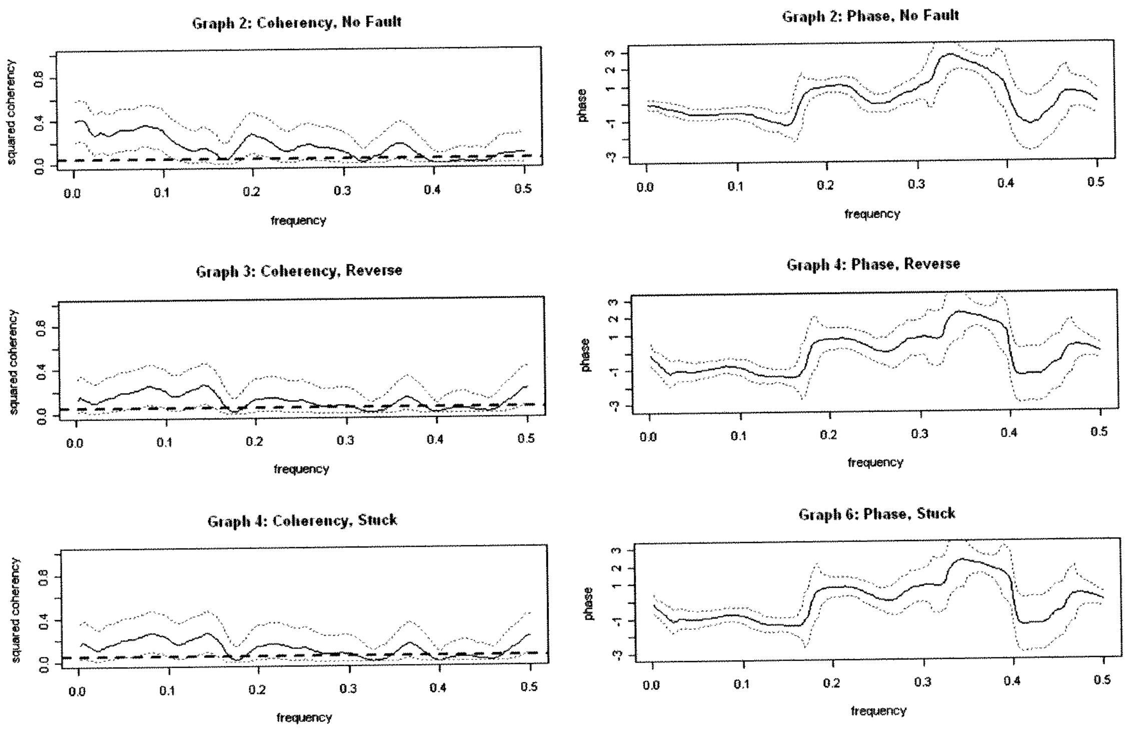


Figure 13. The coherency and phase between the residual and different estimations of MAT for data in Figure 6

Conclusion

The assumption of linear correlation between the heating/ cooling system and the discharge air temperature, and also the simplifications made in the mixing box operation seem to be $\overset{\text{useful}}{\text{justifiable}}$ for RTU diagnostics. It was shown that the mal-functionality of the cooling system can be $\overset{\text{inferred}}{\text{detected}}$ and isolated by monitoring the linear correlation between the cooling stages and DAT variations.

On the issue of the mixing box, the constraint of mixed air temperature measurement was addressed by using a filtered version of the discharge air temperature in which the effect of heating/ cooling system was removed, and the residual was used as a rough estimation of the mixed air temperature. It was shown that the estimated mixed air temperature can be used to analyze the mixing box performance, although due to simplifications and filtering, the confidence level was lower comparing to the case of the heating/ cooling system.

Unclear what you have in mind.

Reference

[1] S. Katipamula, M. R. Brambley, "Methods for fault detection, diagnostics, and prognostics for building systems – a review part I", HVAC&R Research, 2005, vol. 11, n1.

[2] P. Haves, M. Kim, M. Najafi, P. Xu, "A Semi-automated Commissioning Tool for VAV Air Handling Units: Functional Test Analyzer" ASHRAE Transactions. 113, Pt 1. 2007

[3] F. P. Incropera, D. P. DeWitt, T. L. Bergman, A. S. Lavine "Introduction to Heat Transfer", fifth edition, Wiley 2007

[4] R. H. Shumway, D. S. Stoffer, "Time Series Analysis and its Applications with R Examples", second edition, Springer 2005

[5] D. R. Brillinger, "Time Series Data Analysis and Theory", SIAM 2001.

```
rm(list = ls())
setwd("C:/Documents and Settings/Matt/Desktop/Massieh/Courses/stat248/project")

#data1 = read.table("Machine_1638_2008831_2.csv", sep = ",", header = TRUE)
#data = data.frame(data1[,3], data1[,4], data1[,5], data1[,6], data1[,7], data1[,8],
data1[,9], data1[,10], data1[,11])
#rm(data1)
#names(data) = c("rat","dat","oat","dmp", "dmpf","fan", "c1", "c2", "c3")

#dat2 = ts(data[,'dat'])
#oat2 = ts(data[,'oat'])
#rat2 = ts(data[,'rat'])
#c12 = ts(data[,'c1'])
#c22 = ts(data[,'c2'])
#c32 = ts(data[,'c3'])
#dmp2 = ts(data[,'dmp'])
#fan2 = ts(data[,'fan'])



data1 = read.table("Machine_56.csv", sep = ",", header = TRUE)
rat2 = data1[,3]
dat2 = data1[,4]
oat2 = data1[,5]
dmp2 = data1[,7]
c12 = data1[,8]
c22 = data1[,9]
fan2 = data1[,15]
c32 = data1[,16]


j = 0;
dat3=0;oat3=0;rat3=0;c13=0;c23=0;dmp3=0;fan3=0; c33=0
for(k in 1:length(fan2))
{
    if(fan2[k] == 1)
    {
        j = j+1; dat3[j]=dat2[k]; oat3[j]=oat2[k]; rat3[j]=rat2[k]; c13[j]=c12[k]
        c23[j]=c22[k]; c33[j]=c32[k]; dmp3[j]=dmp2[k]; fan3[j]=fan2[k]
    }
}


c1ns=c13;c2ns=c23; c3ns=c33; dmp=dmp3; fan=fan3
datns=dat3; oatns=oat3; ratns=rat3;

str=1
end = 820
#end = 636
hh = 1:length(datns)
windows()
par(mfrow=c(2,2))
```

```
plot(hh[str:end],datns[str:end],type = "l",col="red",ylab="Deg. F", xlab="Time",lwd = 1,
ylim=c(min(datns),max(oatns)))
lines(hh[str:end],oatns[str:end], col="blue", pch=1, lty=1, type = "l")
lines(hh[str:end],ratns[str:end], col="green", pch=1, lty=1, type = "l")
legend(x="topright",c("DAT","OAT", "RAT"), col=c("red", "blue", "green"), lty=1,
     lwd = 2, x.intersp = 0.6, y.intersp = 0.7, cex = 0.8, ncol = 1)
plot(hh[str:end],c1ns[str:end],type = "l",col="red",ylab="On-Off", xlab="Time", lwd = 1,
ylim=c(0,1.15))
legend(x="topright",c("CL1"), col=c("red"), lty=1,
     lwd = 2, x.intersp = 0.7, y.intersp = 0.7, cex = 0.6, ncol = 1)
plot(hh[str:end],c2ns[str:end], col="blue", pch=1, lty=1, type = "l", lwd = 1,
ylab="On-Off", xlab="Time", ylim=c(0,1.15))
legend(x="topright",c("CL2"), col=c("blue"), lty=1,
     lwd = 2, x.intersp = 0.7, y.intersp = 0.7, cex = 0.6, ncol = 1)
plot(hh[str:end],c3ns[str:end], col="black", pch=1, lty=1, type = "l", , ylab="On-Off &
%Open", xlab="Time",lwd =1, ylim=c(0,1.15))
lines(hh[str:end],dmp[str:end]/100, col="green", pch=1, lty=1, type = "l", lwd = 2)
legend(x="topright",c("CL3", "Damper"), col=c("black", "green"), lty=1,
     lwd = 2, x.intersp = 0.7, y.intersp = 0.7, cex = 0.6, ncol = 1)


str=1
end = 1600
hh = 1:length(datns)
windows()
par(mfrow=c(2,1))
plot(hh[str:end],datns[str:end],type = "l",col="red",ylab="F", xlab="Time",lwd = 1,
ylim=c(min(datns),max(oatns)))
lines(hh[str:end],oatns[str:end], col="blue", pch=1, lty=1, type = "l")
lines(hh[str:end],ratns[str:end], col="green", pch=1, lty=1, type = "l")
legend(x="topright",c("DAT","OAT", "RAT"), col=c("red", "blue", "green"), lty=1,
     lwd = 2, x.intersp = 0.6, y.intersp = 0.7, cex = 0.75, ncol = 1)
plot(hh[str:end],c1ns[str:end],type = "l",col="red",ylab="On-Off", xlab="Time", lwd = 3,
ylim=c(0,1.4))
lines(hh[str:end],c2ns[str:end], col="blue", pch=1, lty=1, type = "l", lwd = 1)
lines(hh[str:end],c3ns[str:end], col="black", pch=1, lty=1, type = "l", lwd =2)
legend(x="topright",c("CL1","CL2", "CL3"), col=c("red", "blue", "black"), lty=1,
     lwd = 2, x.intersp = 0.7, y.intersp = 0.7, cex = 0.6, ncol = 1)


ctns = c1ns+c2ns+c3ns

str=1
end = 820
hh = 1:length(datns)
windows()
par(mfrow=c(2,1))
plot(hh[str:end],datns[str:end],type = "l",col="red",ylab="Deg. F", xlab = "Time", lwd =
1, ylim=c(min(datns),max(oatns)))
lines(hh[str:end],oatns[str:end], col="blue", pch=1, lty=1, type = "l")
lines(hh[str:end],ratns[str:end], col="green", pch=1, lty=1, type = "l")
legend(x="topright",c("DAT","OAT", "RAT"), col=c("red", "blue", "green"), lty=1,
     lwd = 2, x.intersp = 0.6, y.intersp = 0.7, cex = 0.8, ncol = 1)
plot(hh[str:end],ctns[str:end],type = "l",col="blue",ylab="On-Off", xlab = "Time", lwd =
2, ylim=c(0,3))
legend(x="topright",c("CT"), col=c("blue"), lty=1,
     lwd = 2, x.intersp = 0.7, y.intersp = 0.7, cex = 0.6, ncol = 1)


c12ns = c1ns+c2ns
c13ns = c1ns+c3ns
c23ns = c2ns+c3ns
matns = (dmp/100)*oatns+(1 - (dmp/100))*ratns
matrns = (1 - (dmp/100))*oatns+(dmp/100)*ratns
matstns = (70/100)*oatns+(1 - (70/100))*ratns
ct = diff(ctns, differences=2)
c1 = diff(c1ns, differences=2)
c2 = diff(c2ns, differences=2)
c3 = diff(c3ns, differences=2)
c12 = diff(c12ns, differences=2)
c13 = diff(c13ns, differences=2)
c23 = diff(c23ns, differences=2)
dat = diff(datns, differences=2)
oat = diff(oatns, differences=2)
rat=diff(ratns, differences=2)
mat = diff(matns, differences=2)
matr = diff(matrns, differences=2)
matst = diff(matstns, differences=2)


datzm = dat - mean(dat)
ratzm = rat - mean(rat)
oatzm = oat - mean(oat)
c1zm = c1 - mean(c1)
c2zm = c2 - mean(c2)
c3zm = c3 - mean(c3)
ctzm = ct - mean(ct)
c12zm = c12 - mean(c12)
c13zm = c13 - mean(c13)
c23zm = c23 - mean(c23)


str=1
#end = 820
end = 636
hh = 1:length(datns)
windows()
par(mfrow=c(2,2))
plot(hh[str:end],dat[str:end],type = "l",col="blue",ylab="Deg. F", xlab = "Time",
  lwd = 1, ylim=c(min(dat),max(dat)), main='DAT difference')
#lines(hh[str:end],oatns[str:end], col="blue", pch=1, lty=1, type = "l")
#lines(hh[str:end],ratns[str:end], col="green", pch=1, lty=1, type = "l")
legend(x="topright",c("DAT"), col=c("blue"), lty=1,
     lwd = 2, x.intersp = 0.6, y.intersp = 0.7, cex = 0.8, ncol = 1)
```

```
acf(dat, main = "DAT difference, Autocorrelation")
plot(hh[str:end],oat[str:end],type = "l",col="blue",ylab="On-Off", xlab = "Time",
   lwd = 2, ylim=c(min(oat),max(oat)), main="OAT difference")
legend(x="topright",c("OAT"), col=c("blue"), lty=1,
     lwd = 2, x.intersp = 0.7, y.intersp = 0.7, cex = 0.6, ncol = 1)
acf(oat, main = 'OAT difference, Autocorrelation')

dat_ct = ts(cbind(datzm,ctzm))
dat_c12 = ts(cbind(datzm,c12zm))
dat_c13 = ts(cbind(datzm,c13zm))
dat_c23 = ts(cbind(datzm,c23zm))
dat_c3 = ts(cbind(datzm,c3zm))

#k = kernel("modified.daniell",c(17,17))
k = kernel("modified.daniell",c(11,11))
windows()
str=1
end = 820
#end = 637
hh = 1:length(dat)
par(mfrow=c(3,2))
plot(hh[str:end], ctns[str:end],type = "l",col="red", lwd = 2, main = 'Graph 1: No
Fault', xlab = 'Time',
  ylab = 'CT', ylim = c(0,3))
st = spec.pgram(dat_ct, k, taper=0.2, detrend = FALSE, demean = FALSE, plot.type="coh",
main='Graph 2: Coherency')
alfa1=0.01
f=qf(1-alfa1,2,st$df-2)
c=f/(2*k$m+1-1+f)
abline(h=c,lty=2)
plot(hh[str:end], c13ns[str:end],type = "l",col="red", lwd = 2, main='Graph 3: CL2
Broken', xlab = 'Time',
    ylab = 'CT', ylim = c(0,3))
s23 = spec.pgram(dat_c3, k, taper=0.2, detrend = FALSE, demean = FALSE, plot.type="coh",
main='Graph 4: Coherency')
alfa1=0.01
f=qf(1-alfa1,2,s23$df-2)
c=f/(2*k$m+1-1+f)
abline(h=c,lty=2)
plot(hh[str:end], c3ns[str:end],type = "l",col="red", lwd = 2, main='Graph 5: CL1 & CL2
Broken', xlab = 'Time',
    ylab = 'CT', ylim = c(0,3))
s13 = spec.pgram(dat_c3, k, taper=0.2, detrend = FALSE, demean = FALSE, plot.type="coh",
main='Graph 6: Coherency')
alfa1=0.01
f=qf(1-alfa1,2,s13$df-2)
c=f/(2*k$m+1-1+f)
abline(h=c,lty=2)


#M = 90  # Number of estimates
M = 48  # Number of estimates
```

```
#k = kernel("daniell", (L-1)/2)
#k = kernel("modified.daniell",c(18,18))
#k = kernel("modified.daniell",c(18,18))
#cc = ctzm
cc = c23zm
#s = spec.pgram(ts(cbind(datzm,ctzm)), k, taper=0.2, plot = FALSE, detrend = FALSE,
demean = FALSE)
s = spec.pgram(ts(cbind(datzm,cc)), k, taper=0.2, plot = FALSE, detrend = FALSE, demean =
FALSE)
fr = s$freq
N= 2*length(fr)
dat.spec = s$spec[,1]
c.spec = s$spec[,2]
coh = s$coh
phase = s$phase


par(mfrow=c(2,2))
spec.pgram(ts(datzm), k, taper=0.2, detrend = FALSE, demean = FALSE, col = 'blue', lwd=2,
main="DAT -- Soothed Periodgram")
spec.pgram(ts(ctzm), k, taper=0.2, detrend = FALSE, demean = FALSE, col = 'blue', lwd=2,
main="CT -- Soothed Periodgram")
alfa1=.01
f=qf(1-alfa1,2,s$df-2)
c=f/(2*k$m+1-1+f)
spec.pgram(ts(cbind(datzm,cc)), k, taper=0.2, detrend = FALSE, demean =
FALSE,plot.type="coh"
  , lwd=2, main="DAT & CT-- Squared Coherency")
abline(h=c,lty=2)
spec.pgram(ts(cbind(datzm,cc)), k, taper=0.2, detrend = FALSE, demean =
FALSE,plot.type="phase"
  , lwd=2, main="DAT & CT-- Phase")
#ccf(datzm,cc, col = 'blue', lwd =2, main="DAT & CT-- Cross-correlation")


##############################
#alfa1=.001
#alfa2=.010
#alfa3=.1
#x=ts(cbind(t.one,t.ten))
#x.spec=spec.pgram(x,k1,taper=.2)
#f=qf(1-alfa1,2,x.spec$df-2)
#c=f/(2*k1$m+1-1+f)
#plot(x.spec,plot.type="coh")
#abline(h=c,lty=2)

############################33


sampled.indices = (N/M)*(1:(M/2))
fr.M = fr[sampled.indices]
coh.M = coh[sampled.indices]
```

```
phase.M = phase[sampled.indices]
dat.spec.M = dat.spec[sampled.indices]
c.spec.M = c.spec[sampled.indices]

B = sqrt(coh*dat.spec/c.spec)*exp(1i*phase)
B.M = sqrt(coh.M*dat.spec.M/c.spec.M)*exp(1i*phase.M)

delta.M = 1/M
Omega.M = seq(from = 1/M, to = .5, length = M/2)
bb.M = function(s) 2*delta.M*sum(exp(2i*pi*Omega.M*s)*B.M)

S.M = ((-M/2+1):(M/2-1))
b.M = vector(length = length(S.M))
for(k in 1:length(S.M)) b.M[k] = bb.M(S.M[k])

windows()
plot(S.M, Re(b.M), type = 'o', col = 'blue', main = 'estimated B for M')
abline(h=0.7, col = 'red')
abline(h=-0.7, col = 'red')

coef = Re(b.M)
alpha1 = (M-2)/2
alpha2 = (M-2)/2 +1
datgen2 = 0
for(i in 1:length(c1))
{
coef3 = (abs(coef)>0.7)*coef
ser = (max(i-alpha1,1)-i): (min(i+alpha1,length(cc))-i)
datgen2[i] = sum(coef3[alpha2-ser]*cc[i+ser])
}

matgen = datzm-datgen2
mm = 1
#pp = 820
pp = 636
windows()
par(mfrow=c(2,2))
plot(S.M, Re(b.M), type = 'o', col = 'blue', main = 'Graph 1: Estimated {B}', xlab='Lag')
abline(h=0.7, col = 'red', lty=3)
abline(h=-0.7, col = 'red', lty=3)
#plot((mm:pp),datzm[mm:pp],type = "l",col="blue", lwd = 1, main='Graph2'
#  , ylab="Deg. F", xlab="Time")
#lines((mm:pp),datgen2[mm:pp], col="red", pch=1, lty=1, type = "l")
#legend(x="topright",c("DAT Original","DAT Constructed"), col=c("blue", "red"), lty=1,
#     lwd = 2, x.intersp = 0.6, y.intersp = 0.7, cex = 0.7, ncol = 1)
plot((mm:pp),matgen[mm:pp],type = "l",col="blue", lwd = 1, ylim=c(-20,20), main="Graph 2:
Residual"
    , ylab="Deg. F", xlab="Time")
legend(x="topright",c("Residual"), col=c("blue"), lty=1,
     lwd = 2, x.intersp = 0.6, y.intersp = 0.7, cex = 0.7, ncol = 1)
acf(matgen, main="Graph 3: Autocorrelation of residual")
yy = range(matns,matrns,matstns)
```

```
plot((mm:pp),matns[mm:pp],type = "l",col="red",ylab="Deg.F", lwd = 1, ylim = yy,
  main = "Graph 4: Mixed Air Temperature(MAT)", xlab="Time")
lines((mm:pp),matrns[mm:pp], col="blue", pch=1, lty=1, type = "l")
lines((mm:pp),matstns[mm:pp], col="black", pch=1, lty=1, type = "l")
legend(x="topright",c("No Fault","Reverse", "Stuck"), col=c("red", "blue", "black"),
lty=1,
     lwd = 1, x.intersp = 0.7, y.intersp = 0.7, cex = 0.7, ncol = 1)




matzm = mat - mean(mat)
matrzm = matr - mean(matr)
matstzm = matst - mean(matst)


windows()
par(mfrow=c(3,2))
#plot((mm:pp),matgen[mm:pp],type = "l",col="black",ylab="Deg.F", lwd = 1, ylim=c(-20,20),
#   main = "Graph1: Mixed Air Temperature(MAT)", xlab="Time")
#lines((mm:pp),matzm[mm:pp], col="red", pch=1, lty=1, type = "l")
#lines((mm:pp),matrzm[mm:pp], col="blue", pch=1, lty=1, type = "l")
#lines((mm:pp),matstzm[mm:pp], col="green", pch=1, lty=1, type = "l")
#legend(x="topright",c("Estimated","No Fault","Reverse", "Stuck"), col=c("black", "red",
"blue", "green"), lty=1,
#     lwd = 1, x.intersp = 0.7, y.intersp = 0.7, cex = 0.7, ncol = 1)


#k = kernel("modified.daniell",c(25,25))
k = kernel("modified.daniell",c(12,12))
s2 = spec.pgram(ts(cbind(matgen,matzm)), k, taper=0.2, plot = FALSE, detrend = FALSE,
demean = FALSE)
#s2 = spec.pgram(ts(cbind(matgenfl,matzmfl)), k, plot = FALSE, detrend = FALSE, demean =
FALSE)
fr = s2$freq
coh2 = s2$coh

alfa1=.1
f=qf(1-alfa1,2,s2$df-2)
c=f/(2*k$m+1-1+f)
plot(s2,plot.type="coh", main="Graph 2: Coherency, No Fault")
abline(h=c,lty=2, lwd=2)
plot(s2,plot.type="phase", main="Graph 2: Phase, No Fault")


#s3 = spec.pgram(ts(cbind(matgenfl,matrzmfl)), k, taper=0.2, plot = FALSE, detrend =
FALSE, demean = FALSE)
s3 = spec.pgram(ts(cbind(matgen,matrzm)), k, taper=0.2, plot = FALSE, detrend = FALSE,
demean = FALSE)
coh3 = s3$coh
alfa1=.1
f=qf(1-alfa1,2,s3$df-2)
```

```
c=f/(2*k$m+1-1+f)
plot(s3,plot.type="coh", main="Graph 3: Coherency, Reverse")
abline(h=c,lty=2, lwd=2)
plot(s3,plot.type="phase", main="Graph 4: Phase, Reverse")

#k = kernel("modified.daniell",c(12,12))
#s4 = spec.pgram(ts(cbind(matgenfl,matstzmfl)), k, taper=0.2, plot = FALSE, detrend =
FALSE, demean = FALSE)
s4 = spec.pgram(ts(cbind(matgen,matstzm)), k, taper=0.2, plot = FALSE, detrend = FALSE,
demean = FALSE)
coh4 = s4$coh
alfa1=.1
f=qf(1-alfa1,2,s4$df-2)
c=f/(2*k$m+1-1+f)
plot(s4,plot.type="coh", main="Graph 4: Coherency, Stuck")
#plot(s4,plot.type="phase")
abline(h=c,lty=2, lwd=2)
plot(s4,plot.type="phase", main="Graph 6: Phase, Stuck")


#library(TSA)
windows()
par(mfrow=c(2,2))
prewhiten(matgen,matzm, ylim=c(-0.5,0.5), col='blue', main="Cross-correlation, no fault")
prewhiten(matgen,matrzm, ylim=c(-0.5,0.5), col='blue', main="Cross-correlation, reverse")
prewhiten(matgen,matstzm, ylim=c(-0.5,0.5), col='blue', main="Cross-correlation, stuck")
```