Grey-box modeling of the heat dynamics of a building with CTSM-R

Rune Juhl Niels Rode Kristensen Peder Bacher Jan Kloppenborg
Henrik Madsen

August 3, 2022

Chapter 1

Introduction

This document is an example of using the R package CTSM-R for grey-box modeling of the heat dynamics of a building. A two-state model is implemented and the results are analyzed. Then the two-state model is extended to a three-state model, which is fitted and analyzed, and compared to the two-state model with a likelihood ratio test, in order to determine if the three-state model is a more suitable model compared to the two-state model. The data and the two models are taken from the case-study presented in Bacher and Madsen (2011).

1.1 Initiate

To start the modeling a few initialization steps are carried out. Note that here the working directory needs to be to the path where the files are located on the computer:

```
## Init by deleting all variables and functions
rm(list=ls())

## Set the working directory. Change this to the location of the example on the computer.
setwd(".")

## Use the CTSM-R package, note that first the package must be installed, see the Installa
library(ctsmr)

## List with global parameters
prm <- list()
## Number of threads used by CTSM-R for the estimation computations
prm$threads <- 1
```

Then source some functions defined in the files in the `functions` folder:

```
## Source the scripts with functions in the "functions" folder. Just a neat way of arrangi
sapply(dir("functions", full.names=TRUE), source)
```

1.2 Read the data

The data used in this example was measured in an office building which is part of the smart-grid experimental facility SYSLAB is DTU Elektro, Risø campus laboratory for intelligent distributed power systems [1]. The building is built in a lightweight wood construction. The time series consist of five-minutes averaged values of:

[1] `www.powerlab.dk/English/facilities/SysLab.aspx`

- T_i (`yTi` in data) the average of all the indoor temperatures measured (one in each room in the building). The sensors were hanging approximately in the center of each room (°C).
- Φ_h (`Ph` in data) the total heat input for all electrical heaters in the building (kW).
- T_a (`Ta` in data) the ambient temperature (°C).
- G (`Ps` in the data) the global radiation (W/m^2).
- W_s (`Ws` in data) the wind speed (m/s).

The climate variables were measured with a climate station right next to the building. See Bacher and Madsen (2010) for more details of the experiments in which the data was recorded.
The data is located in the file `inputPRBS1.csv` which are read into a `data.frame` by:

```
## Read the data into a data.frame
X <- read.csv("inputPRBS1.csv",sep=";",header=TRUE)
## X$t is now hours since start of the experiment. Create a column in the POSIXct format f
X$timedate <- asP("2009-02-05 14:26:00)") + X$t * 3600
```

Plot the time series. Two helping functions (found in the `functions` folder) are used for setting up the plot of the data in Figure 1.1.

```r
## Plot the time series (see "functions/setpar.R" to see the plot setup function)
setpar("ts", mfrow=c(4,1))
gridSeq <- seq(asP("2009-01-01"),by="days",len=365)
##
plot(X$timedate,X$yTi,type="n",xlab="",ylab="yTi ($^{\\circ}$C)")
abline(v=gridSeq,h=0,col="grey85",lty=3)
lines(X$timedate,X$yTi)
##
plot(X$timedate,X$Ta,type="n",xlab="",ylab="Ta ($^{\\circ}$C)")
abline(v=gridSeq,h=0,col="grey85",lty=3)
lines(X$timedate,X$Ta)
##
plot(X$timedate,X$Ph,type="n",xlab="",ylab="Ph (kW)")
abline(v=gridSeq,h=0,col="grey85",lty=3)
lines(X$timedate,X$Ph)
##
plot(X$timedate,X$Ps,type="n",xlab="",ylab="Ps (kw/m$^2$)")
abline(v=gridSeq,h=0,col="grey85",lty=3)
lines(X$timedate,X$Ps)
axis.POSIXct(1, X$timedate, xaxt="s", format="%Y-%m-%d")
```

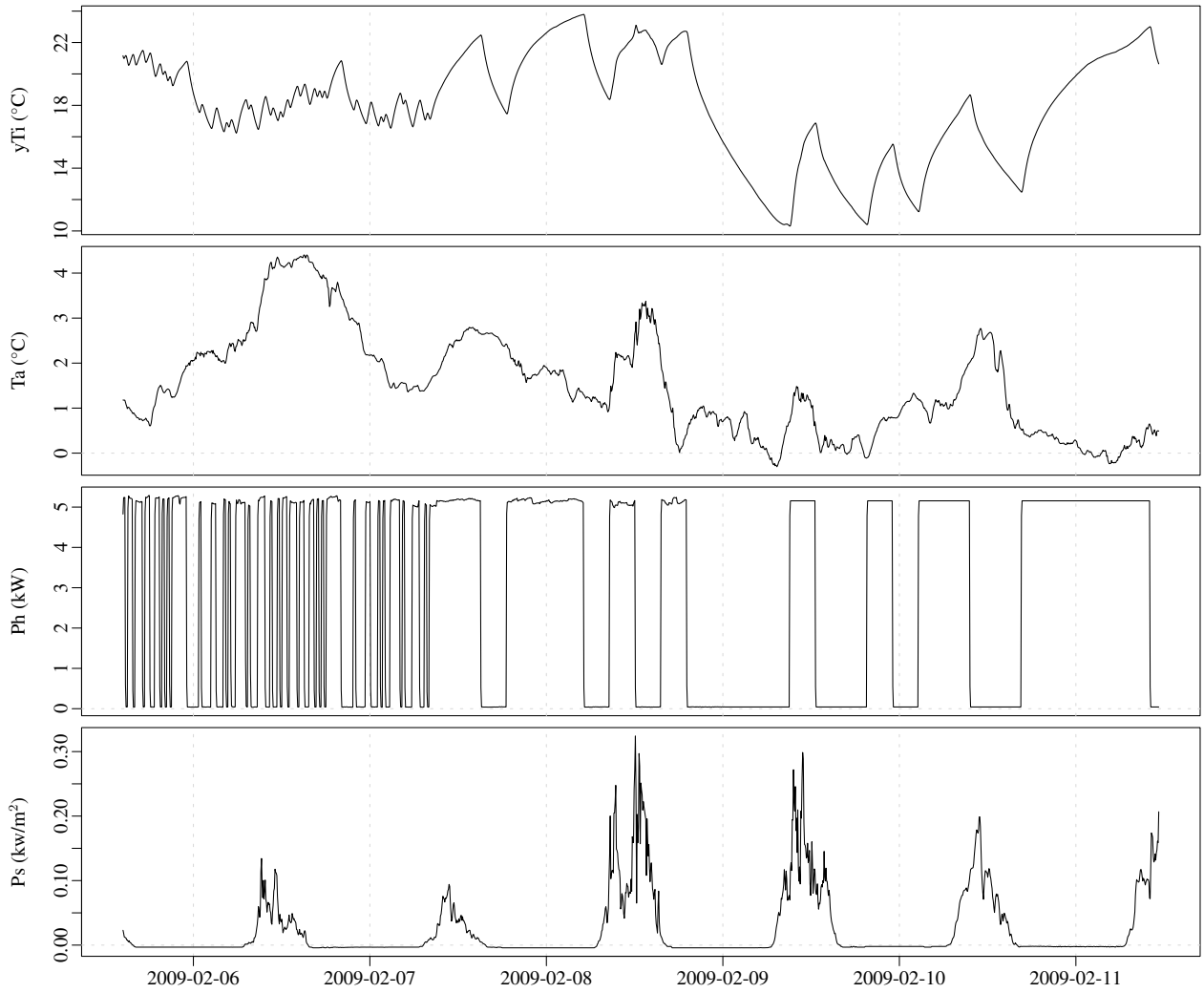


Figure 1.1: Plots of the data.

1.3 Two-state grey-box model of the heat dynamics of a building

The two-state grey-box model *TiTe* illustrated with the RC-diagram in Figure 1.2 and defined by the system equations

$$dT_\mathrm{i} = \Big(\frac{1}{R_\mathrm{ia}C_\mathrm{i}}(T_\mathrm{e}-T_\mathrm{i}) + \frac{1}{R_\mathrm{ih}C_\mathrm{i}}(T_\mathrm{h}-T_\mathrm{i}) + \frac{1}{C_\mathrm{i}}A_\mathrm{w}\Phi_\mathrm{s} + \frac{1}{C_\mathrm{i}}\Phi_\mathrm{h}\Big)dt + \sigma_\mathrm{i}d\omega_\mathrm{i} \tag{1.1}$$

$$dT_\mathrm{e} = \Big(\frac{1}{R_\mathrm{ie}C_\mathrm{e}}(T_\mathrm{i}-T_\mathrm{e}) + \frac{1}{R_\mathrm{ea}C_\mathrm{e}}(T_\mathrm{a}-T_\mathrm{e})\Big)dt + \sigma_\mathrm{e}d\omega_\mathrm{e} \tag{1.2}$$

together with the measurement equation

$$Y_{t_k} = T_{\mathrm{i},t_k} + e_{t_k} \tag{1.3}$$

is specified with CTSM-R in the following.

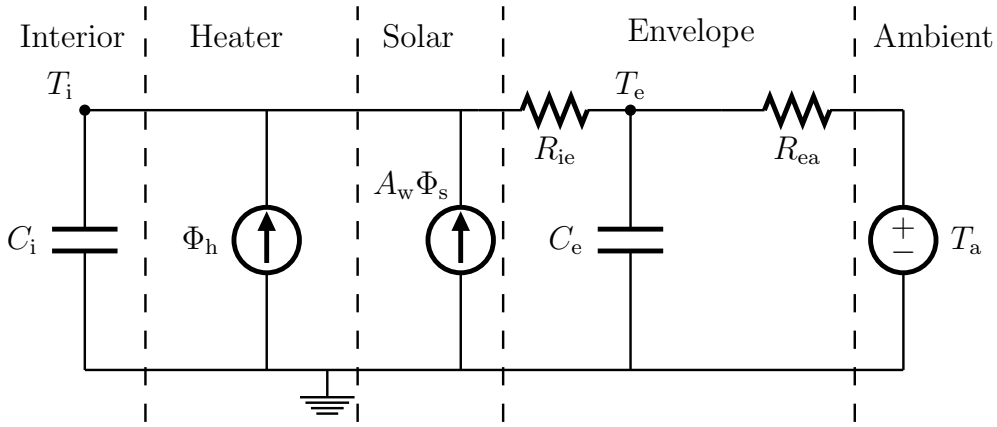


Figure 1.2: RC-network network of *TiTe*.

First, a model object is generated. Then the two state equations are added and the inputs defined:

```
## Generate a new object of class ctsm
model <- ctsm$new()
## Add system equations and thereby also states
model$addSystem(dTi ~ ( 1/(Ci*Rie)*(Te-Ti) + Aw/Ci*Ps + 1/Ci*Ph )*dt + exp(p11)*dw1)
model$addSystem(dTe ~ ( 1/(Ce*Rie)*(Ti-Te) + 1/(Ce*Rea)*(Ta-Te) )*dt + exp(p22)*dw2)
## Set the names of the inputs
model$addInput(Ta,Ps,Ph)
```

Note the following for each equation

- the deterministic part of the SDE is multiplied with `dt`,
- the stochastic part is multiplied with system noise process `dw1`,
- the system noise is scaled with `exp(p11)`, where `exp()` is the exponential function and `p11` is the parameter which is estimated. Using the exponential function is since variance (and likewise standard deviation) is strictly positive, but can be very close to zero, it is advised to take the exponential function, in order to get a better numerical resolution in the optimization. See Section for a detailed explanation of the interpretation of the noise level parameters.

The observation equation is added:

```
## Set the observation equation: Ti is the state, yTi is the measured output
model$addObs(yTi ~ Ti)
## Set the variance of the measurement error
model$setVariance(yTi ~ exp(e11))
```

Set the intial values of the states and parameters together with bounds for the optimization:

```
## Set the initial value (for the optimization) of the states at the start time point
model$setParameter(  Ti = c(init=15  ,lb=0     ,ub=25) )
model$setParameter(  Te = c(init=5   ,lb=-20   ,ub=25) )
## Set the initial value of the parameters for the optimization
model$setParameter(  Ci = c(init=1   ,lb=1E-5  ,ub=20) )
model$setParameter(  Ce = c(init=2   ,lb=1E-5  ,ub=20) )
model$setParameter( Rie = c(init=10  ,lb=1E-5  ,ub=50) )
model$setParameter( Rea = c(init=10  ,lb=1E-5  ,ub=50) )
model$setParameter(  Aw = c(init=20  ,lb=0.1   ,ub=200))
model$setParameter( p11 = c(init=1   ,lb=-50   ,ub=10) )
model$setParameter( p22 = c(init=1   ,lb=-50   ,ub=10) )
model$setParameter( e11 = c(init=-1  ,lb=-50   ,ub=10) )
```

Finally, run the parameter estimation:

```
## Run the parameter optimization
fit <- model$estimate(data=X, threads=prm$threads)
```

and keep the results in `fit`.

1.3.1 Model validation

Evaluate the result of the parameter estimation for the two-state model.
First an extended summary of the fit is printed with:

```
## See the summary of the estimation
print(summary(fit,extended=TRUE))

## Coefficients:
##        Estimate  Std. Error     t value    Pr(>|t|)     dF/dPar dPen/dPar
## Ti0  2.1166e+01  4.8672e-02  4.3486e+02  0.0000e+00 -4.9019e-04    0.0036
## Te0 -1.9846e+01  9.0254e-01 -2.1989e+01  0.0000e+00  3.3226e-05    1.6778
## Aw   2.3803e+01  4.2510e+00  5.5994e+00  2.5150e-08  1.8342e-06    0.0000
## Ce   1.4403e+00  1.3489e+00  1.0677e+00  2.8579e-01  1.1301e-06    0.0000
## Ci   5.7188e+00  2.3971e-01  2.3857e+01  0.0000e+00 -1.1886e-05    0.0001
## e11 -2.7224e+01  4.7137e+02 -5.7756e-02  9.5395e-01  2.6076e-06    0.0002
## p11 -1.7531e+01  1.0556e+01 -1.6609e+00  9.6929e-02  6.0008e-05    0.0001
## p22  2.9155e+00  3.3869e-01  8.6084e+00  0.0000e+00 -4.8124e-05    0.0001
## Rea  1.1888e+00  2.1653e+00  5.4902e-01  5.8306e-01  2.0923e-06    0.0000
## Rie  2.4269e+00  7.3688e-01  3.2935e+00  1.0103e-03  1.4955e-05    0.0000
##
## Correlation of coefficients:
##     Ti0   Te0   Aw    Ce    Ci    e11   p11   p22   Rea
## Te0  0.03
## Aw   0.70  0.07
## Ce   0.87 -0.01  0.68
## Ci   0.36  0.05  0.44  0.38
## e11  0.94  0.03  0.76  0.90  0.41
## p11 -0.94 -0.07 -0.76 -0.90 -0.43 -1.00
## p22  0.95 -0.03  0.67  0.87  0.38  0.91 -0.91
## Rea -0.92  0.00 -0.72 -0.98 -0.38 -0.95  0.95 -0.92
## Rie  0.94 -0.04  0.62  0.85  0.25  0.88 -0.88  0.99 -0.90
```

The following four important points are checked (see the section: Model Validation in the CTSM-RUser Guide)

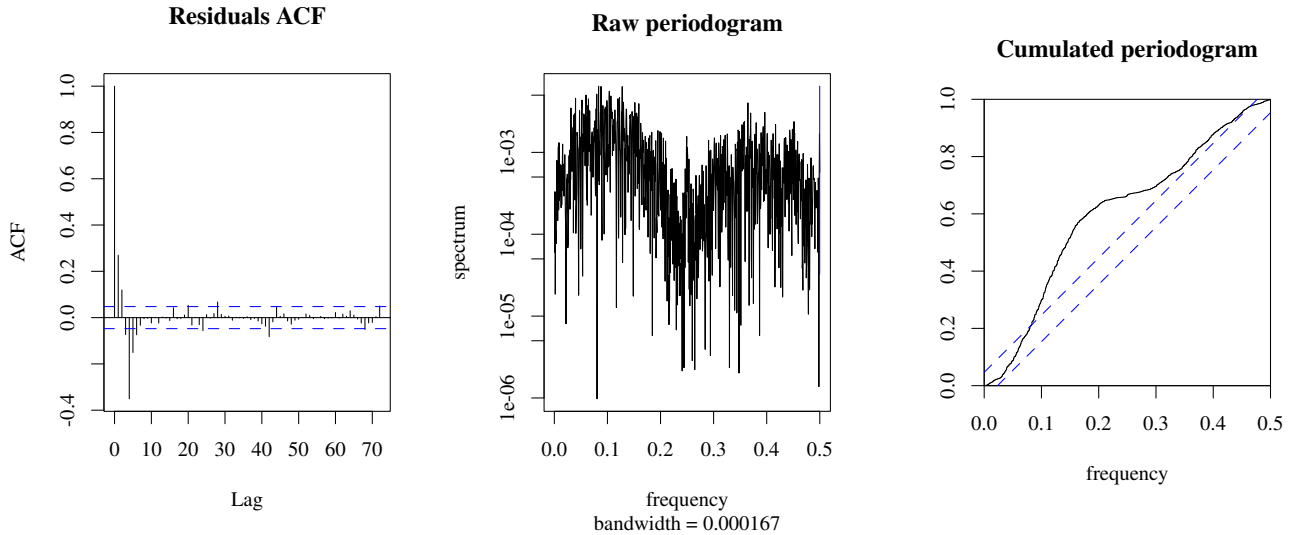
- That the p-value of the t-tests (i.e. `Pr(>|t|)`) is below 0.05 for all parameters
- That the derivative of the objective function with respect to each parameter (i.e. `dF/dPar`) is close to zero
- That the derivative of the penalty function with respect to each parameter (i.e. `dPen/dPar`) is not significant compared to `dF/dPar`). Here it is noticed that the value for `Te0`, the initial value of the state T_e, is much higher than `dF/dPar`. This indicates that the optimization ended at one of the bounds for `Te0` and it is seen that this is the lower bound, which was set to -20 °C. Clearly this is not a realistic initial value of the state T_e representing the temperature of the envelope (the ambient temperature is around 1 °C). Hence, this indicates that the parameter estimates are not optimal for the model.
- That Correlation Matrix do not have any off-diagonal values close to -1 or 1.

The one-step ahead predictions and residuals are then calculated:

```
## Calculate the one-step predictions of the state (i.e. the residuals)
tmp <- predict(fit)[[1]]
## Calculate the residuals and put them with the data in a data.frame X
X$residuals <- X$yTi - tmp$output$pred$yTi
X$yTiHat <- tmp$output$pred$yTi
```

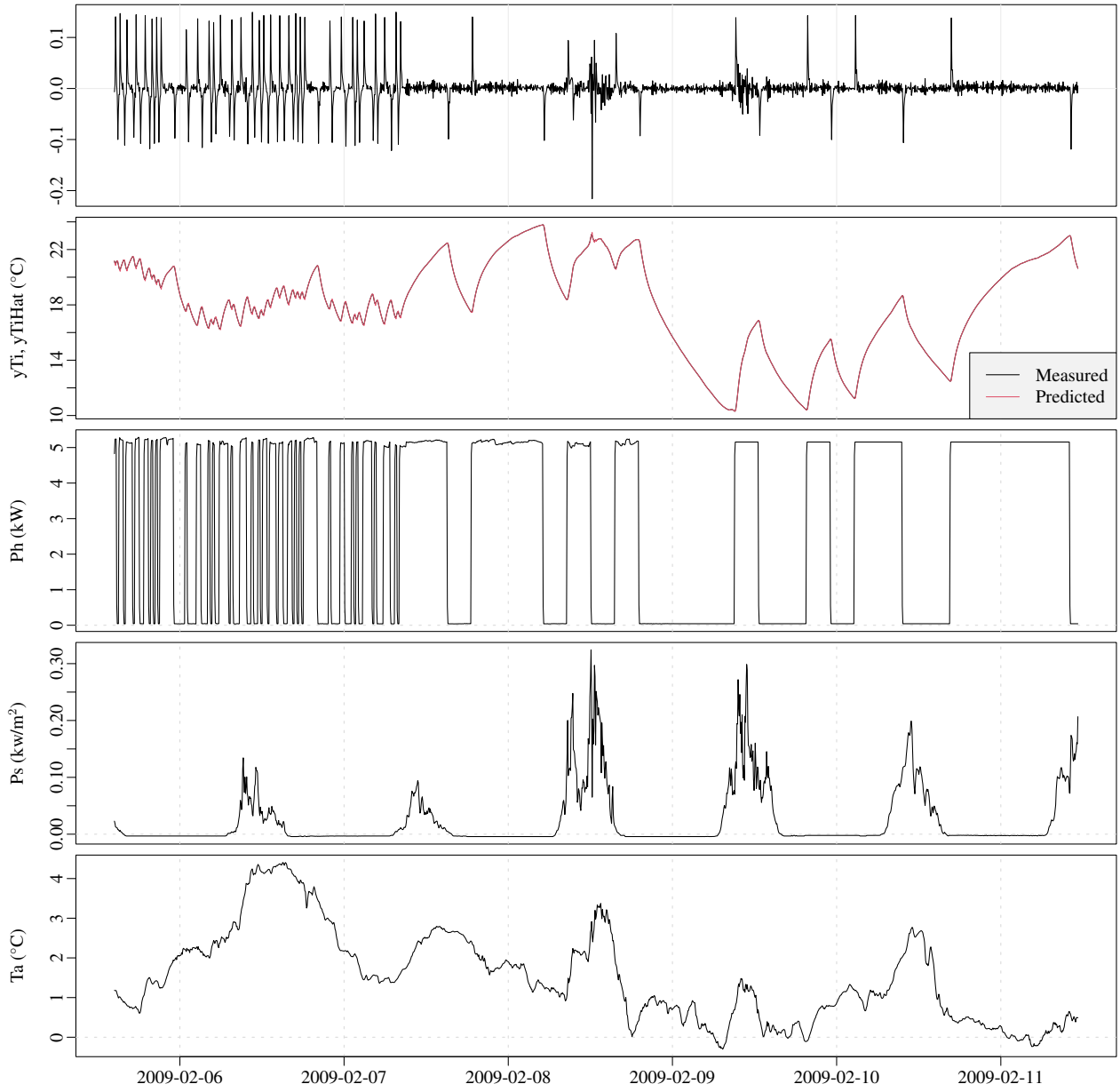
and the auto-correlation function, the periodogram and the cumulated periodogram are plotted:

```
## Plot the auto-correlation function and cumulated periodogram in a new window
par(mfrow=c(1,3))
## The blue lines indicates the 95 confidence interval, meaning that if it is
## white noise, then approximately 1 out of 20 lag correlations will be slightly outside
acf(X$residuals, lag.max=6*12, main="Residuals ACF")
## The periodogram is the estimated energy spectrum in the signal
spec.pgram(X$residuals, main="Raw periodogram")
## The cumulated periodogram
cpgram(X$residuals, main="Cumulated periodogram")
```



Clearly, the residuals are not white noise and it is concluded that the model lacks in the describtion of the heat dynamics of the building. Finally, time series plots of the residuals and the inputs are plotted:

```r
## Plot the time series (see "functions/setpar.R" to see the plot setup function)
setpar("ts", mfrow=c(5,1))
gridSeq <- seq(asP("2009-01-01"), by="days", len=365)
##
plot(X$timedate, X$residuals, xlab="yTi ($^{\\circ}$C)", ylab="", type="n")
abline(v=gridSeq, h=0, col="grey92")
lines(X$timedate, X$residuals)
##
plot(X$timedate, X$yTi, ylim=range(X[ ,c("yTi","yTiHat")]), type="n", xlab="", ylab="yTi,
abline(v=gridSeq, h=0, col="grey85", lty=3)
lines(X$timedate, X$yTi)
lines(X$timedate, X$yTiHat, col=2)
legend("bottomright", c("Measured","Predicted"), lty=1, col=1:2, bg="grey95")
##
plot(X$timedate, X$Ph, type="n", xlab="", ylab="Ph (kW)")
abline(v=gridSeq, h=0, col="grey85", lty=3)
lines(X$timedate, X$Ph)
##
plot(X$timedate, X$Ps, type="n", xlab="", ylab="Ps (kw/m$^2$)")
abline(v=gridSeq, h=0, col="grey85", lty=3)
lines(X$timedate, X$Ps)
##
plot(X$timedate, X$Ta, type="n", xlab="", ylab="Ta ($^{\\circ}$C)")
abline(v=gridSeq, h=0, col="grey85", lty=3)
lines(X$timedate, X$Ta)
axis.POSIXct(1, X$timedate, xaxt="s", format="%Y-%m-%d")
```



Considering the time series plot of the residuals, it becomes apparent that the dynamics of the system is poorly modeled right after level shifts in the PRBS heat input signal, i.e. every time the heaters are turned on and off. In the two-state model *TiTe* the heat from the heaters is flowing directly into indoor air and the thermal inertia of the heaters is not taken into account. This leads to the idea of including a part in the model which represent the heaters, as carried out in the three-state model described in the following section.

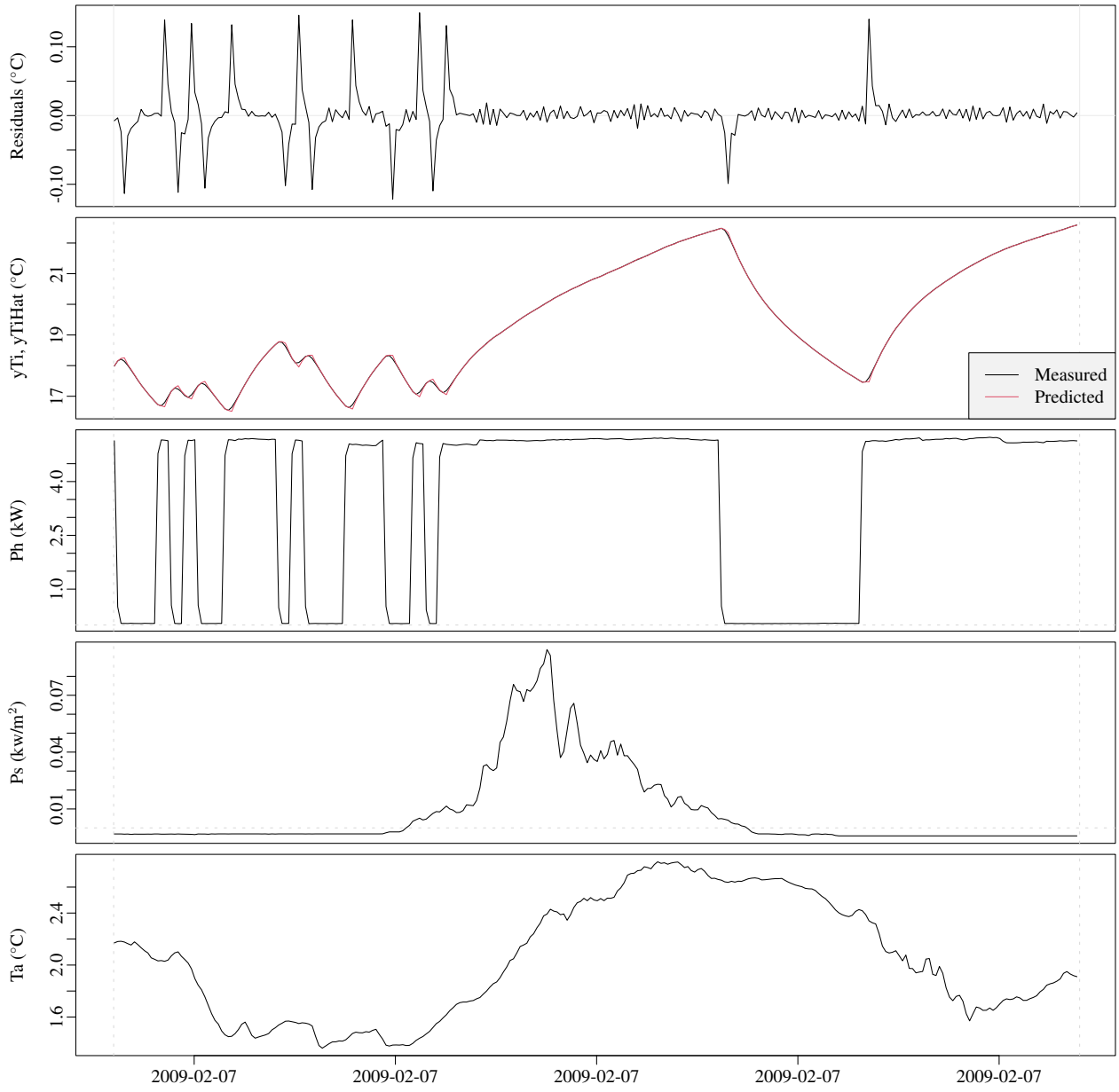
1.4 Wrapping the model and validation into functions

Now, in order to organize the code to achieve a more efficient work flow, the two-state model is implemented in a function `TiTe(Dat)`, which is defined in the file `functions/TiTe.R`. Open the file to see how the function is defined. In the function everything needed are specified and the model is fitted to the data given as the argument `Dat`. The function is executed with:

```
fitTiTe <- TiTe(X)
```

where the list returned by the function includes the model, the fit, and the data.
Furthermore, the code for analyzing and validating the fit is implemented in a function `analyzeFit()` (see the file "`functions/analyzeFit.R`"). The list returned by the model function is given as an argument:

```
analyzeFit(fitTiTe ,tPer=c("2009-02-07","2009-02-08"),plotACF=FALSE)
```



and here also the argument `tPer` is set in order to only analyze a given period of one day. From the plots of the residuals it is easily seen that when the heaters are switched on and off, some fast dynamics are not described well by the model.

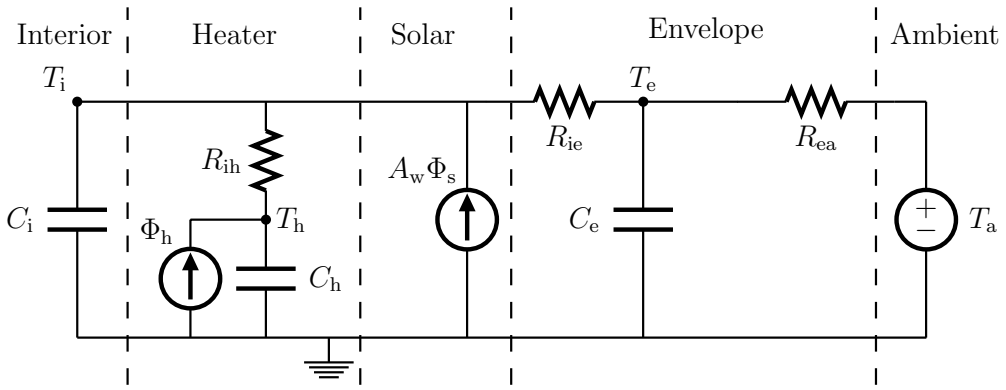


Figure 1.3: RC-network network of the three-state model *TiTeTh*.

1.5 Three-state model

In this section a three-state model is fitted to the data and the results are analyzed. The model is an extension of the two-state, where a heat capacitor and a thermal resistance are added to represent the heaters in the building. A temperature state T_e representing the temperature of the heaters is included and the model is

$$dT_\mathrm{i} = \Big(\frac{1}{R_\mathrm{ia}C_\mathrm{i}}(T_\mathrm{e} - T_\mathrm{i}) + \frac{1}{R_\mathrm{ih}C_\mathrm{i}}(T_\mathrm{h} - T_\mathrm{i}) + \frac{1}{C_\mathrm{i}}A_\mathrm{w}\Phi_\mathrm{s}\Big)dt + \sigma_\mathrm{i}d\omega_\mathrm{i} \tag{1.4}$$

$$dT_\mathrm{e} = \Big(\frac{1}{R_\mathrm{ie}C_\mathrm{e}}(T_\mathrm{i} - T_\mathrm{e}) + \frac{1}{R_\mathrm{ea}C_\mathrm{e}}(T_\mathrm{a} - T_\mathrm{e})\Big)dt + \sigma_\mathrm{e}d\omega_\mathrm{e} \tag{1.5}$$

$$dT_\mathrm{h} = \Big(\frac{1}{R_\mathrm{ih}C_\mathrm{h}}(T_\mathrm{i} - T_\mathrm{h}) + \frac{1}{C_\mathrm{h}}\Phi_\mathrm{h}\Big)dt + \sigma_\mathrm{e}d\omega_\mathrm{h} \tag{1.6}$$

and the measurement equation is

$$Y_{t_k} = T_{\mathrm{i},t_k} + e_{t_k} \tag{1.7}$$

The RC-diagram in Figure 1.3 is illustrating the model and it is denoted *TiTeTh*.
Like the two-state model as described in Section 1.4, the three-state model is wrapped in a function:

```
fitTiTeTh <- TiTeTh(X)
```

where the list returned by the function includes the model, the fit, and the data.
Now, the estimation results of can easily be analyzed, as for the two-state model in Section 1.4 with the same function. The summary is printed:

```
analyzeFit(fitTiTeTh,plotACF=FALSE,plotSeries=FALSE)

## Coefficients:
##        Estimate  Std. Error     t value    Pr(>|t|)     dF/dPar dPen/dPar
## Ti0  2.1159e+01  5.8444e-03  3.6204e+03  0.0000e+00 -5.4692e-03    0.0036
## Te0  1.8262e+01  2.0850e-01  8.7587e+01  0.0000e+00 -5.7220e-05    0.0010
## Th0 -1.3228e+01  3.8251e+01 -3.4581e-01  7.2953e-01 -8.6945e-06    0.0063
## Aw   5.6346e+00  2.7422e-01  2.0548e+01  0.0000e+00  1.6835e-04    0.0000
## Ce   2.9169e+00  1.3904e-01  2.0979e+01  0.0000e+00  3.5916e-05    0.0000
## Ch   1.0867e-03  1.5203e-03  7.1480e-01  4.7483e-01  7.6114e-04    0.0000
## Ci   1.0723e+00  1.1947e-02  8.9756e+01  0.0000e+00  1.1718e-03    0.0000
## e11 -1.2891e+01  8.9527e-01 -1.4399e+01  0.0000e+00 -8.2009e-05    0.0000
## p11 -4.0795e+00  1.5214e-01 -2.6813e+01  0.0000e+00 -9.4743e-06    0.0000
## p22 -1.3170e+00  4.6879e-02 -2.8093e+01  0.0000e+00  7.8620e-05    0.0000
## p33 -4.5741e+00  3.2440e+00 -1.4100e+00  1.5872e-01 -9.0953e-06    0.0000
## Rea  4.5389e+00  9.6151e-02  4.7206e+01  0.0000e+00  9.8179e-05    0.0000
## Rie  8.6312e-01  2.3894e-02  3.6123e+01  0.0000e+00  2.7080e-04    0.0000
## Rih  1.1904e+02  1.6642e+02  7.1528e-01  4.7453e-01  7.8442e-04    0.0000
```

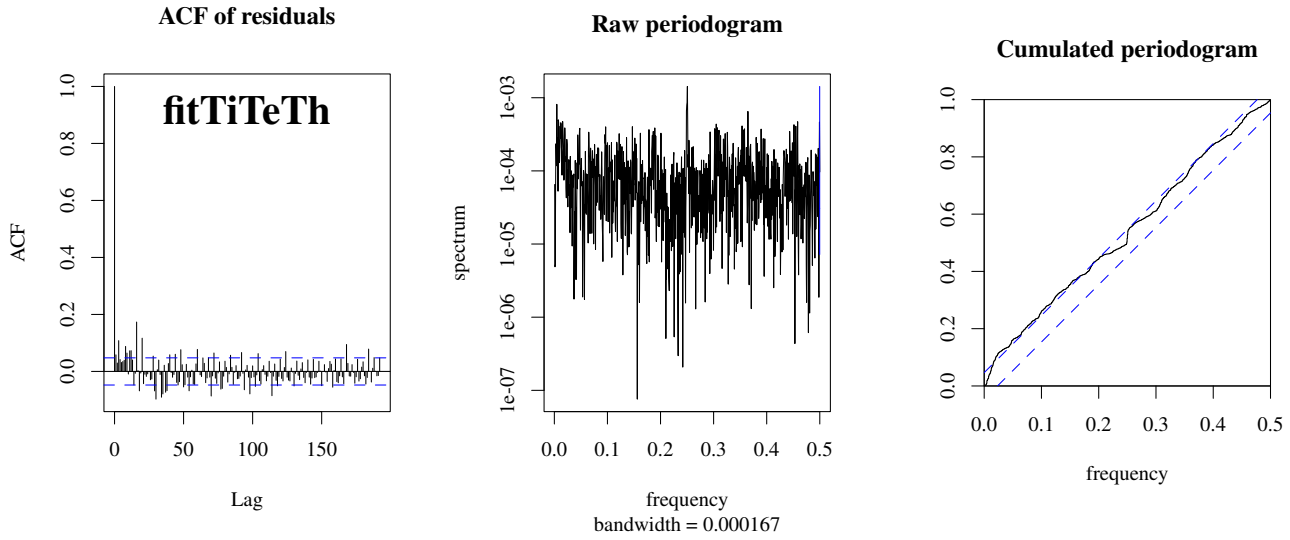
```
##
## Correlation of coefficients:
##      Ti0   Te0   Th0   Aw    Ce    Ch    Ci    e11   p11   p22   p33   Rea
## Te0  0.20
## Th0  0.02  0.06
## Aw  -0.01 -0.14 -0.12
## Ce   0.00 -0.12 -0.05  0.12
## Ch   0.05  0.55  0.74 -0.13 -0.05
## Ci   0.01 -0.39 -0.19  0.22  0.24 -0.15
## e11 -0.03  0.14 -0.02 -0.16 -0.12  0.02 -0.19
## p11  0.03 -0.16 -0.01  0.21  0.15 -0.04  0.25 -0.96
## p22  0.01 -0.13 -0.03 -0.06 -0.03  0.04  0.26  0.33 -0.38
## p33 -0.01  0.12 -0.97  0.11  0.05 -0.56  0.19  0.02  0.00  0.04
## Rea  0.04  0.08  0.00 -0.10 -0.06 -0.04 -0.21  0.05 -0.07 -0.11 -0.02
## Rie  0.01 -0.43 -0.08  0.18  0.22 -0.04  0.69 -0.17  0.22  0.45  0.09 -0.27
## Rih -0.05 -0.55 -0.74  0.13  0.05 -1.00  0.14 -0.02  0.04 -0.04  0.56  0.05
##      Rie
## Te0
## Th0
## Aw
## Ce
## Ch
## Ci
## e11
## p11
## p22
## p33
## Rea
## Rie
## Rih  0.03
##
## [1] "Loglikelihood 5516.38449472404"
```

As for the two-state model fit the following four important points are checked (see the section: Model Validation in the CTSM-RUser Guide)

- That the p-value of the t-tests (i.e. `Pr(>|t|)`) is below 0.05 for all parameters, except for the estimate of `Th0` and `p33`, but this is found to be ok, since they can be considered as nuisance parameters (i.e. helping parameters not considered in conclusions. Further, the `p` and `e` are taken through the exp).
- That the derivative of the objective function with respect to each parameter (i.e. `dF/dPar`) is close to zero
- That the derivative of the penalty function with respect to each parameter (i.e. `dPen/dPar`) is not significant compared to `dF/dPar`).
- That Correlation Matrix do not have any off-diagonal values close to -1 or 1, except between `p22` and `Ch`, which is found to be fine, since are not very important parameters.

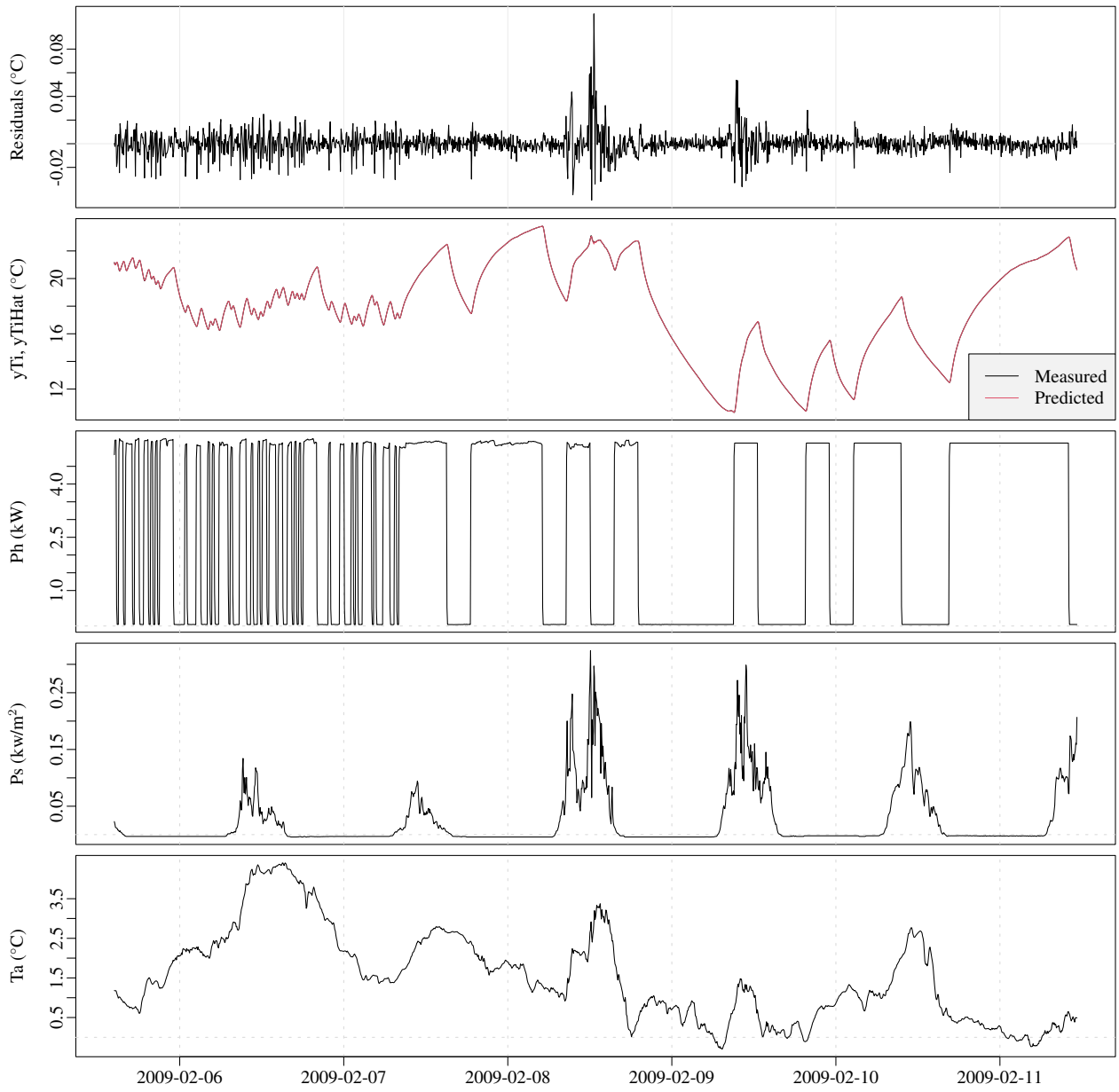
The auto-correlation and cumulated periodogram is plotted:

```
analyzeFit(fitTiTeTh,plotSeries=FALSE)
```



and clearly, the one-step ahead prediction residuals are much alike white noise, especially compared to the residuals for the two-state model.
Finally, the plots of the time series:

```
analyzeFit(fitTiTeTh,plotACF=FALSE)
```



confirms that the residuals are much closer to white noise. Studying the residuals a bit closer reveals that the 8'th and 9'th, have some shorter periods with direct solar radiation, and that the level of the residuals in these periods is increased. Therefore further expansion of the model could be focused on improving the part of the model, where the solar radiation enters the building.

1.6 Estimated total HLC-value of the building

The estimated total HLC-value (heat loss coefficient, possibly heat transfer coefficient (HTC) or UA-value) of the building is for the three state model simply calculated by

$$\widehat{HLC} = \frac{1}{\hat{R}_{\text{ie}} + \hat{R}_{\text{ea}}} \tag{1.8}$$

However the estimate of its standard deviance is a bit harder to calculate. It is a non-linear function of two normal distributed values, so two reasonable approaches can be applied: a linear approximation and

a simulation approach. First the HLC-value is calculated and then the covariance of the two R estimates are taken:

```
## The estimated HLC-value
i <- which(names(fitTiTeTh$xm)%in%c("Rea","Rie"))
HLC <- 1/sum(fitTiTeTh$xm[i])
HLC*1000 ## W/C

## [1] 185.1151

## The covariance for the two estimated R values
cov <- diag(fitTiTeTh$sd[i]) %*% fitTiTeTh$corr[i,i] %*% diag(fitTiTeTh$sd[i])
```

For the linear approximation the Jacobian is calculated and used to calculate the variance of the estimated HLC-value:

```
## The Jacobian, the derived of the HLC-value with respect to each estimate in fitTiTeTh$x
J <- t( sapply(1:length(i), function(ii,x){ -1/sum(x)^2 }, x=fitTiTeTh$xm[i]) )
## The estimated variance of U
varHLC <- J %*% cov %*% t(J)
## and standard deviance
sdHLC <- sqrt(varHLC)
## Return the confidence interval
c(HLC-1.96*sdHLC, HLC+1.96*sdHLC)*1000

## [1] 178.8931 191.3371
```

For the simulation approach a set of multivariate normal values based on the estimates of the R values are generated. The HLC-value is then calculated for the generated values and the 2.5% and 97.5% quantiles are estimated as the confidence interval:

```
## Needed for multivariate normal distribution simulation
require(MASS)

## Loading required package:  MASS

## Generate multivariate normal random values
Rsim <- mvrnorm(n=1000000, mu=fitTiTeTh$xm[i], Sigma=cov)
## For each realization calculate the HLC-value
HLCsim <- 1/apply(Rsim, 1, sum)
## Estimate the 2.5% and 97.5% quantiles of the simulated values as a confidence interval
quantile(HLCsim, probs=c(0.025,0.975))*1000

##     2.5%    97.5%
## 179.1023 191.5386
```

1.7 Likelihood-ratio test

Likelihood-ratio tests are very useful for determining whether a larger model is to be preferred over smaller model (i.e. the smaller model is a submodel of the larger model), see Madsen and Thyregod (2010). Here a likelihood ratio test comparing the likelihood of the two-state model to the likelihood of the three-state model:

```r
## Take the results of both models
small <- fitTiTe
large <- fitTiTeTh
## Calculate the logLikelihood for both models from their fit
logLikSmallModel <- small$loglik
logLikLargeModel <- large$loglik
## Calculate lambda
chisqStat <- -2 * (logLikSmallModel - logLikLargeModel)
## It this gives a p-value smaller than confidence limit, i.e. 5\%, then the
## larger model is significant better than the smaller model
prmDiff <- large$model$NPARAM - small$model$NPARAM
## The p-value of the test
1 - pchisq(chisqStat, prmDiff)

## [1] 0
```

The p-value is very low and thus the three-state model is preferred over the two-state model.

Bibliography

P. Bacher and H. Madsen. Experiments and data for building energy performance analysis : Financed by the danish electricity saving trust. Technical report, DTU Informatics, Building 321, Kgs. Lyngby, 2010.

P. Bacher and H. Madsen. Identifying suitable models for the heat dynamics of buildings. *Energy & Buildings*, 43(7):1511–1522, 2011. ISSN 03787788. doi: 10.1016/j.enbuild.2011.02.005.

H. Madsen and P. Thyregod. *Introduction to General and Generalized Linear Models*. CRC Press, 2010.

J. K. Møller and H. Madsen. From state dependent diffusion to constant diffusion in stochastic differential equations by the lamperti transform, 2010.

Appendix A

A.1 Saving a fit

In order to save a fit and use it again for analysis, it is needed to save the model alongside. After loading the fit and the model in a new Rprocess (i.e. after restarting R), it is needed to recompile the model, this is necessary since CTSM-Rcompiles Fortran code for the model functions.
Simply save the fit and the model with:

```
save(fitTiTeTh,file="fitTiTeTh.rda")
```

and then in a new Rprocess load the fit again:

```
load("fitTiTeTh.rda")
```

A.2 Noise level estimates and interpretation

In the R formulation of the models the `exp(p11)` is the estimate of σ_{i}, which is a scaling of the system noise. It has the units: Units of state variable over square root time units.

The Wiener process is defined such that the variance increases with the time Δt between two observations

$$\omega_{t+\Delta t} - \omega_t = \Delta\omega \sim N(0, \Delta t) \tag{A.1}$$

Hence the variance of the system noise, which is added to each state variable in the model, is

$$\mathrm{V}(\sigma\Delta\omega) = \sigma^2\,\mathrm{V}(\Delta\omega) = \sigma^2\Delta t \tag{A.2}$$

Thus

$$\sigma\Delta\omega \sim N(0, \sigma^2\Delta t) \tag{A.3}$$

For the presented data the time is in hours and the time unit is hour. The states are temperatures, hence the units are

$$[\,\mathrm{V}(\Delta\omega)] = \mathrm{h} \Leftrightarrow \tag{A.4}$$

$$[\Delta\omega] = \sqrt{\mathrm{h}} \tag{A.5}$$

leading to

$$[\sigma_x] = \frac{^{\circ}\mathrm{C}}{\sqrt{\mathrm{h}}} \tag{A.6}$$

which is not easily interpreted.

But it means that the variance system noise process for the *TiTeTh* in Equation 1.4, for each state per hourly step, is estimated to

$$\mathrm{V}(\hat{\sigma}_{\mathrm{i}}\Delta\omega_{\mathrm{i}}) = (e^{-4.08}\frac{^{\circ}\mathrm{C}}{\sqrt{\mathrm{h}}})^2 1\mathrm{h} = (0.0169\ ^{\circ}\mathrm{C})^2 \tag{A.7}$$

$$\mathrm{V}(\hat{\sigma}_{\mathrm{e}}\Delta\omega_{\mathrm{e}}) = (e^{-1.32}\frac{^{\circ}\mathrm{C}}{\sqrt{\mathrm{h}}})^2 1\mathrm{h} = (0.267\ ^{\circ}\mathrm{C})^2 \tag{A.8}$$

$$\mathrm{V}(\hat{\sigma}_{\mathrm{h}}\Delta\omega_{\mathrm{h}}) = (e^{-4.57}\frac{^{\circ}\mathrm{C}}{\sqrt{\mathrm{h}}})^2 1\mathrm{h} = (0.010\ ^{\circ}\mathrm{C})^2 \tag{A.9}$$

Now this is the estimated variance "per hour", so to get a sense of the system noise level compared to the level of measurement noise, this has to be related to the 5 minutes steps by a sum of 12 independent random variables, so

$$\sum_{i=1}^{12} \Delta\omega_{5\mathrm{min}} = \Delta\omega \tag{A.10}$$

and the variance

$$\mathrm{V}\left(\sum_{i=1}^{12} \Delta\omega_i^{5\mathrm{min}}\right) = \mathrm{V}(\Delta\omega) \Leftrightarrow \tag{A.11}$$

$$12\ \mathrm{V}(\Delta\omega_{5\mathrm{min}}) = \mathrm{V}(\Delta\omega) \Leftrightarrow \tag{A.12}$$

$$\mathrm{V}(\Delta\omega_{5\mathrm{min}}) = \frac{\mathrm{V}(\Delta\omega)}{12} \tag{A.13}$$

For the three states this results in standard deviations of the process

$$0.0169\ ^{\circ}\mathrm{C}/\sqrt{12} = 0.0049\ ^{\circ}\mathrm{C} \tag{A.14}$$

$$0.267\ ^{\circ}\mathrm{C}/\sqrt{12} = 0.077\ ^{\circ}\mathrm{C} \tag{A.15}$$

$$0.010\ ^{\circ}\mathrm{C}/\sqrt{12} = 0.0029\ ^{\circ}\mathrm{C} \tag{A.16}$$

which can be compared directly to the estimated standard deviation of measurement noise

$$\sqrt{e^{-12.9}} = 0.0016\ ^{\circ}\mathrm{C} \tag{A.17}$$

i.e. Equation (1.7) holds the random variable $e_{t_k} \sim N\big(0, (0.0016\ ^{\circ}\mathrm{C})^2\big)$ representing the measurement noise.

In conclusion the level of system noise is estimated to be magnitudes higher with the *TiTeTh* - of course, formulating a more accurate model should decrease the system noise level, but not the measurement noise level.

The standard deviation of the one-step predictions of the states are calculated by:

```
val <- predict(fitTiTeTh)[[1]]
head(val$state$sd)
```

```
##             Ti          Te          Th
## 1 0.006042697 0.07588682 0.002757023
## 2 0.008393796 0.09793233 0.003480913
## 3 0.008642389 0.10021230 0.003596157
## 4 0.008653234 0.10030323 0.003616983
## 5 0.008653517 0.10030539 0.003622325
## 6 0.008653521 0.10030542 0.003623809
```

which are slightly higher than the values above, since they are predictions - so state estimates plus the system noise.
One should get the system noise variance level by:

```
valfilt <- filter.ctsmr(fitTiTeTh)[[1]]

tail(val$state$sd)

##                Ti        Te          Th
## 1686 0.008653522 0.1003054 0.003624379
## 1687 0.008653522 0.1003054 0.003624379
## 1688 0.008653522 0.1003054 0.003624379
## 1689 0.008653522 0.1003054 0.003624379
## 1690 0.008653522 0.1003054 0.003624379
## 1691 0.008653522 0.1003054 0.003624379

tail(valfilt$sd)

##                Ti        Te          Th
## 1686 0.001561357 0.0681181 0.002860527
## 1687 0.001561357 0.0681181 0.002860527
## 1688 0.001561357 0.0681181 0.002860527
## 1689 0.001561357 0.0681181 0.002860527
## 1690 0.001561357 0.0681181 0.002860527
## 1691 0.001561357 0.0681181 0.002860527

sqrt(tail(val$state$sd^2 - valfilt$sd^2))

##                Ti         Te          Th
## 1686 0.008511498 0.07362813 0.002225649
## 1687 0.008511498 0.07362813 0.002225649
## 1688 0.008511498 0.07362813 0.002225649
## 1689 0.008511498 0.07362813 0.002225649
## 1690 0.008511498 0.07362813 0.002225649
## 1691 0.008511498 0.07362813 0.002225649
```

but doesn't comply with the values. This must be further checked...!!??
Remember that σ can be a function of the inputs, see CTSM-RUser Guide, but not of the state variables directly - however that can be done using a Lamperti transform as described by Møller and Madsen (2010).