DTU02417, Time Series Analysis, Assignment 1
Steady State Building Model

Edward J. Xu (Jie Xu), s181238, DTU Management, edxu96@outlook.com

March 14th, 2019

1 Question 1, Data Visualization

The data about heating is shown in fig.1. There are fluctuations and increasing trend. Because it's from a stochastic process depending on other processes, it's essential to establish the model with other processes and predict it's value on their predictions.

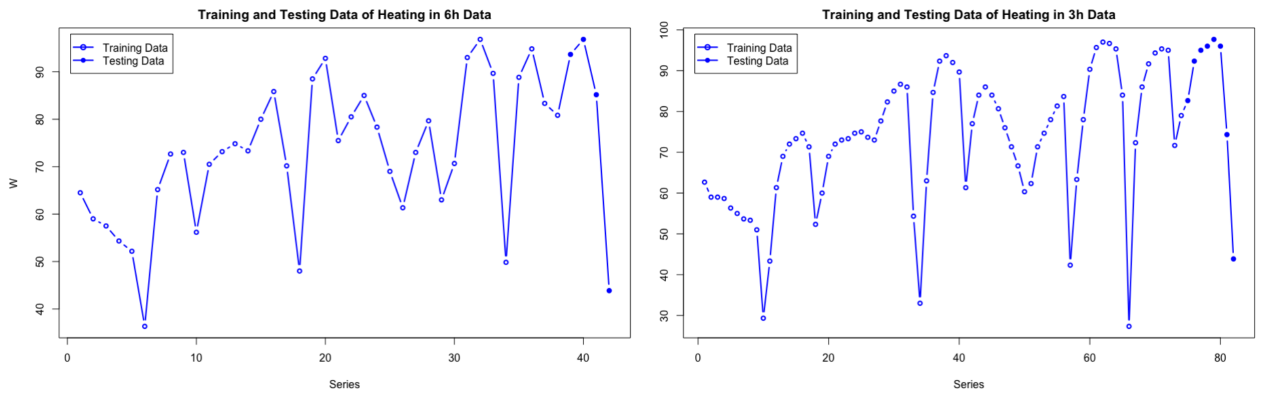

Figure 1. Training and Testing Data of Heating

The data about external temperature is shown in fig.2. The downward trend is obvious, which can be described by linear regression. Although there is no visible curve in the trend, a quadratic trend model can be established to be compared to usual linear trend model. The fluctuations are unstable, there will be large variations in the validation and prediction.

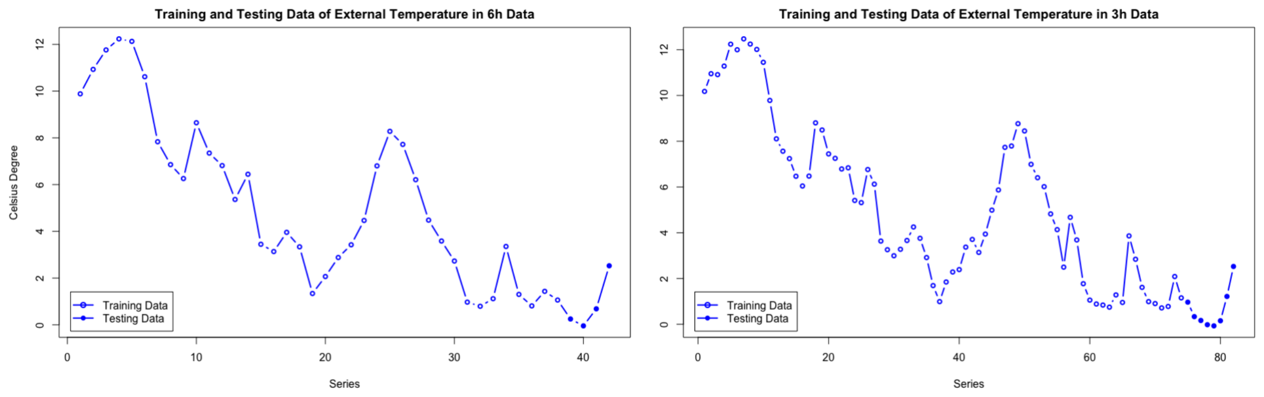

Figure 2. Training and Testing Data of External Temperature

The data about solar irradiation is shown in fig.3. There are regular fluctuations, which can be explained by the regular solar movement. So it can be modeled by seasonal local trend model. Also, there is a bit increasing trend, which can be explained by the time when the data are obtained. The height angle of the sun is increasing, so the solar irradiation in the corresponding time during the day is increasing, with the exception of that during cloudy days.

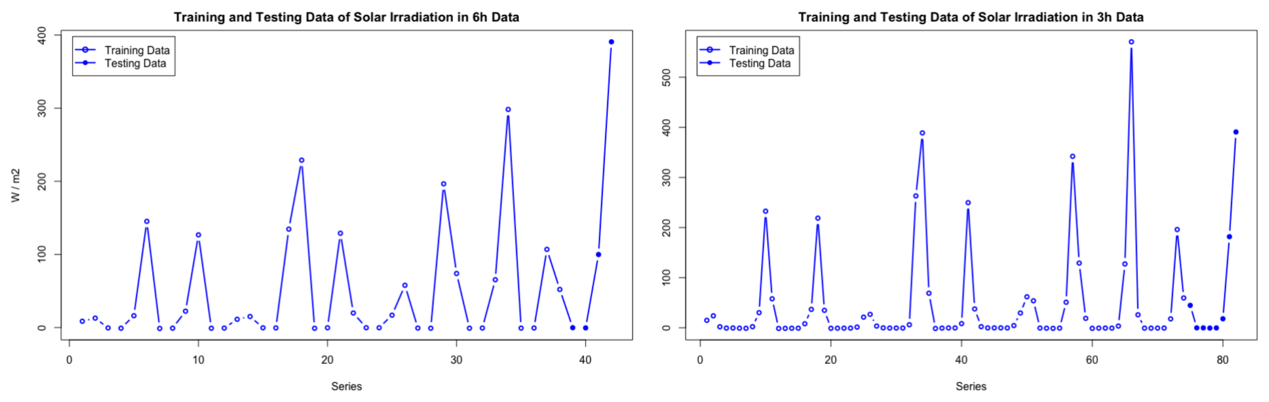

Figure 3. Training and Testing Data of Solar Irradiation

2 Question 2: Weighted Least Square Estimation of General Linear Model (GLM)

Firstly, it is assumed that the residuals have a constant variance and are independent. So the following linear regression model is formulated to estimate $\boldsymbol{\theta}$. The estimation method is ordinary least square in page.36-37 in book.[1].

$$\Phi_{\text{h}} = H_{\text{tot}} T_{\text{in}} - H_{\text{tot}} T_{\text{ex}} + \text{g}A_{\text{sol}} I_{\text{sol}} + \varepsilon \tag{1}$$

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\theta} + \varepsilon \tag{2}$$

$$\mathbf{Y} = (\Phi_{\text{h, 1}}, \Phi_{\text{h, 2}}, \ldots, \Phi_{\text{h, N}})^T \tag{3}$$

$$\mathbf{X} = \begin{pmatrix} 1 & T_{\text{ex},1} & I_{\text{sol},1} \\ 1 & T_{\text{ex},2} & I_{\text{sol},2} \\ \vdots & \vdots & \vdots \\ 1 & T_{\text{ex},N} & I_{\text{sol},N} \end{pmatrix} \tag{4}$$

$$\boldsymbol{\theta} = (H_{\text{tot}} T_{\text{in}}, -H_{\text{tot}}, \text{g}A_{\text{sol}})^T \tag{5}$$

The result is summarized in the following section.2.0.1. This is the plot of residuals from the result.

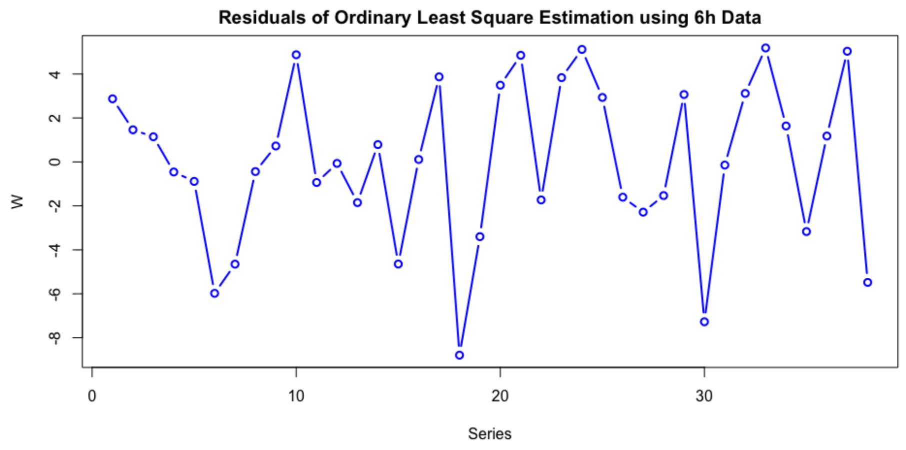

Figure 4. Residuals of Ordinary Least Square Estimation using 6h Data

2.0.1 *Least Square when Residiuals with Exponential Decaying using 6h Data*

Now, we assume that the correlation structure of the residiuals is an exponential decaying function of the time distance between two observations.

$$\text{Cor}\left(\varepsilon_{t_i}, \varepsilon_{t_i}\right) = \rho^{|t_i - t_j|} \quad , 0 < \rho < 1 \tag{6}$$

$$\Sigma = \begin{pmatrix} 1 & \rho & \rho^2 & \cdots \\ \rho & 1 & \rho & \cdots \\ \rho^2 & \rho & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{7}$$

```r
do_mat_capSigma_expDecay <- function(rho, n){
    mat_capSigma <- diag(n)
    for (i in 1: n){
        for (j in 1: n){
            mat_capSigma[i, j] <- rho^(abs(i - j))
        }
    }
    return(mat_capSigma)
}
```

Code 1. R Function to Generate Σ with Exponential Decaying Function with ρ

A relaxation algorithm can be designed to find the optimal value of ρ, which is described in book.[2]. The R code is shown below.

```r
cal_rho_expDecayRelaxAlgo <- function(mat_x, vec_y){
    rho <- 0
    n <- length(mat_x[,1])
    for (t in 1: 5){
        mat_capSigma <- do_mat_capSigma_expDecay(rho, n)
        vec_theta.hat <- pred_vec_theta.hat(mat_x, vec_y, mat_capSigma)
        sigma.hat <- cal_sigma.hat(mat_x, vec_y, vec_theta.hat)
        vec_epsilon <- vec_y - mat_x %*% vec_theta.hat
        rho <- do_newRho(mat_x, vec_epsilon, sigma.hat, n)
    }
    return(rho)
}
```

Code 2. R Function of Relaxation Algorithm for Finding Optimal ρ

The next ρ is determined by:

$$\operatorname{Cor}\left(e_{t_i}, e_{t_j}\right) = \frac{\operatorname{Cov}\left(e_{t_i}, e_{t_j}\right)}{\sigma_{e_{t_i}} \sigma_{e_{t_j}}} = \frac{E\left[\left(e_{t_i} - E\left[e_{t_i}\right]\right)\left(e_{t_j} - E\left[e_{t_j}\right]\right)\right]}{\sigma_{e_{t_i}} \sigma_{e_{t_j}}} \tag{8}$$

$$= \frac{E\left[e_{t_i} e_{t_j}\right]}{\sigma^2} \tag{9}$$

$$= \frac{\frac{1}{N-1} \sum e_t e_{t+1}}{\sigma^2} \tag{10}$$

```r
do_newRho <- function(mat_x, vec_epsilon, sigma, n){
    sum <- 0
    for (i in 1: (n - 1)){
        sum <- sum + vec_epsilon[i] * vec_epsilon[i+1]
    }
    rho <- sum / (sigma^2 * (n - 1))
    return(rho)
}
```

Code 3. R Function in Relaxation Algorithm to Find Next ρ

The result of least square estimation using 6h data when residiuals have exponential decaying is:

$$\hat{\rho} = 0.0533 \tag{11}$$

$$\hat{T}_i = 28.3 \tag{12}$$

The residuals are compared to the result when Sigma is identity matrix.

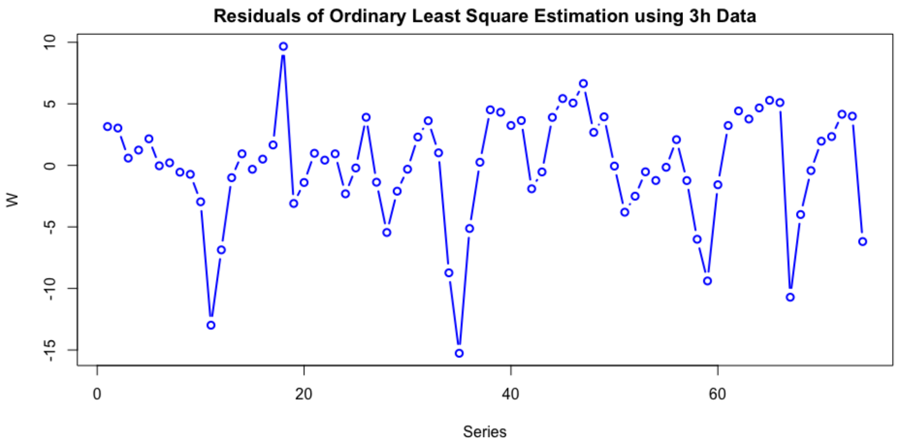

Figure 5. Comparison of Residuals from OLS and WLS(Exp Decay) using 6h Data

The results of two least square estimation using 6h data are summarized in tab.1

Σ	ρ	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\sigma}$	$\hat{T}_i$
Sigma being Identity Matrixt	0	96.34049	-3.404605	-0.1231323	3.748241	28.29711
Sigma with Exp Decaying	0.0533173	96.24318	-3.392797	-0.1224553	3.748756	28.36691

Table 1. Results of Two Least Square Estimation using 6h Data

$$V(\hat{\theta}) = \begin{pmatrix} 1.509988674 & -0.1775305436 & -4.343437e-03 \\ -0.177530544 & 0.0316079776 & 2.125901e-04 \\ -0.004343437 & 0.0002125901 & 7.072094e-05 \end{pmatrix} \text{ when Sigma being Identity Matrixt} \tag{13}$$

$$V(\hat{\theta}) = \begin{pmatrix} 1.510403027 & -0.1775792593 & -4.344629e-03 \\ -0.177579259 & 0.0316166511 & 2.126484e-04 \\ -0.004344629 & 0.0002126484 & 7.074035e-05 \end{pmatrix} \text{ when Sigma with Exp Decaying} \tag{14}$$

2.0.2 Estimation of GLM using 3h Data

The result of ordinary least square estimation using 3h data is:

$$\hat{T}_i = 28.3 \tag{15}$$

The result of least square estimation using 3h data when residiuals have exponential decaying is:

$$\hat{\rho} = 0.0533 \tag{16}$$

$$\hat{T}_i = 28.3 \tag{17}$$

The residuals are compared to the result when Sigma is identity matrix.

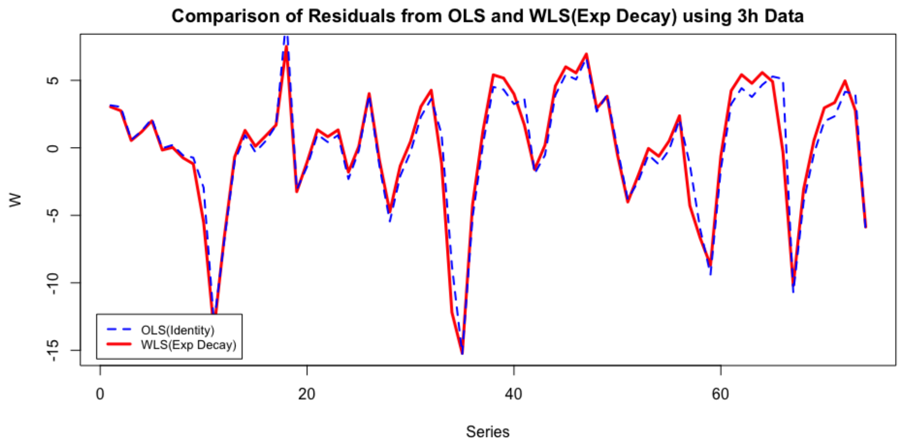

Figure 6. Comparison of Residuals from OLS and WLS(Exp Decay) using 3h Data

The results of two least square estimation using 3h data are summarized in tab.2.

Σ	ρ	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\sigma}$	$\hat{T}_i$
Sigma being Identity Matrixt	0	95.34499	-3.366492	-0.1053378	4.543622	28.32177
Sigma with Exp Decaying	0.5564169	94.26587	-3.264975	-0.0946527	4.688327	28.87185

Table 2. Results of Two Least Square Estimation using 3h Data

$$V(\hat{\theta}) = \begin{pmatrix} 1.013384372 & -1.258880e-01 & -1.427078e-03 \\ -0.125888035 & 2.338349e-02 & 3.645831e-05 \\ -0.001427078 & 3.645831e-05 & 2.678816e-05 \end{pmatrix} \quad \text{when Sigma being Identity Matrixt} \tag{18}$$

$$V(\hat{\theta}) = \begin{pmatrix} 1.078960914 & -1.340343e-01 & -1.519425e-03 \\ -0.134034305 & 2.489665e-02 & 3.881755e-05 \\ -0.001519425 & 3.881755e-05 & 2.852163e-05 \end{pmatrix} \quad \text{when Sigma with Exp Decaying} \tag{19}$$

2.1 Effect of Sampling Size on the Residuals

It's obvious that the residuals of OLS and WLS(Exp Decay) are different when the 3h data are estimated.

3 Question 3, Local Trend Mode (0.8) of Outdoor Temperature using 6h Data

The formulation of the local trend model for external temperature.

$$\mathbf{Y} = (T_{\text{ex},1}, T_{\text{ex},2}, ..., T_{\text{ex},N})^T \tag{20}$$

$$\mathbf{X} = (\mathbf{f}^T(-N+1), \mathbf{f}^T(-N+2), ..., \mathbf{f}^T(0))^T \tag{21}$$

Convergence of total memory of local trend model can be visualized:

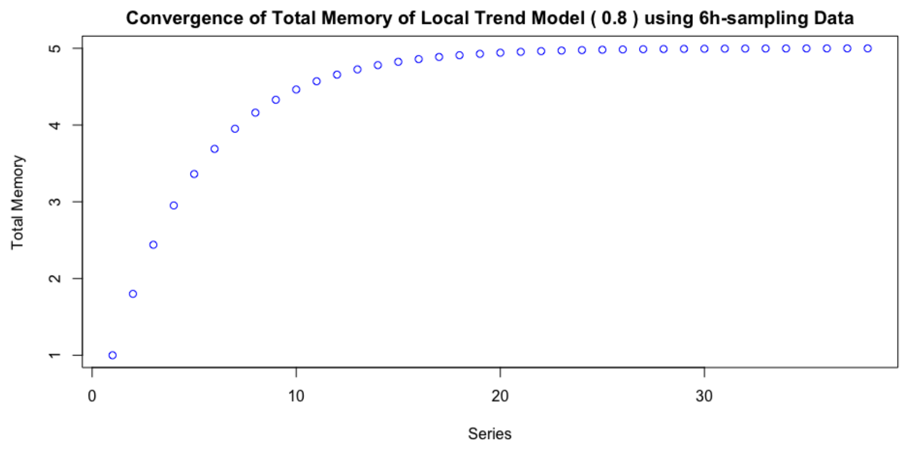

Figure 7. Convergence of Total Memory of Local Trend Model (0.8) using 6h Data

3.1 Linear Local Trend Model (0.8)

Forecast function in linear local trend model is defined:

$$\mathbf{f}(j) = (1, j)^T \tag{22}$$

The result is:

$$\hat{\theta} = (0.8819388, -0.2519854)^T \tag{23}$$

$$\hat{\sigma} = 0.4317291 \tag{24}$$

$$\mathrm{V}\left[\hat{\theta}\right] = \begin{pmatrix} 0.06744921 & 0.00755578 \\ 0.00755578 & 0.00189268 \end{pmatrix} \tag{25}$$

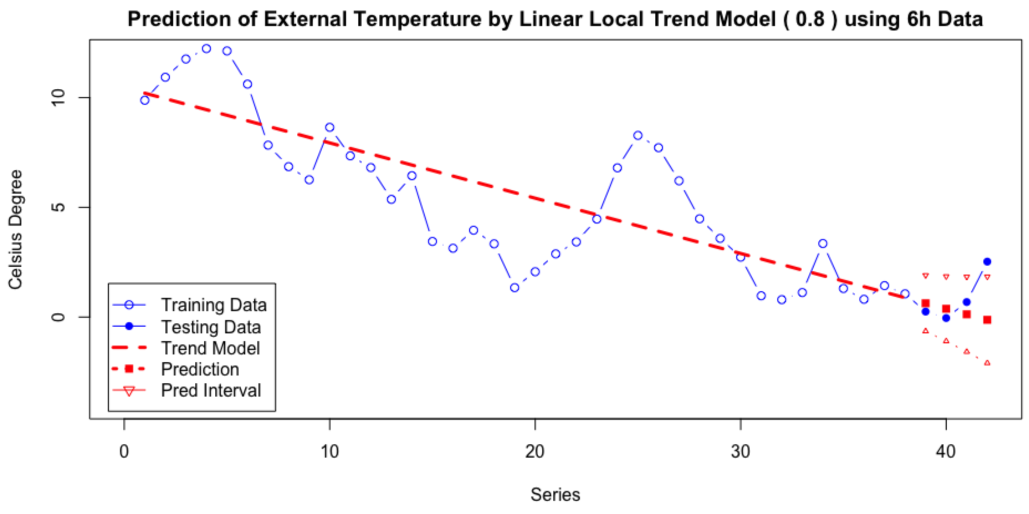

Figure 8. Prediction of External Temperature by Linear Local Trend Model (0.8) using 6h Data

3.2 Quadratic Local Trend Model (0.8)

Forecast function in quadratic local trend model is defined:

$$\mathbf{f}(j) = (1, j, j^2/2)^T \tag{26}$$

The result is:

$$\hat{\theta} = (0.820272643, -0.285539419, -0.004056753)^T \tag{27}$$

$$\hat{\sigma} = 0.4369668 \tag{28}$$

$$V\left[\hat{\theta}\right] = \begin{pmatrix} 0.09584972 & 0.0222976962 & 0.0017600298 \\ 0.02229770 & 0.0098599447 & 0.0009576736 \\ 0.00176003 & 0.0009576736 & 0.0001157848 \end{pmatrix} \tag{29}$$

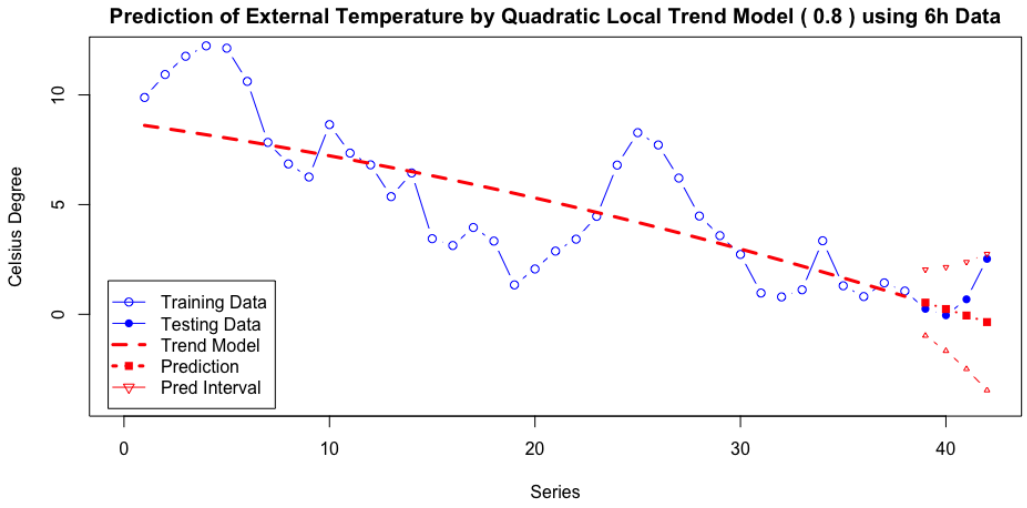

Figure 9. Prediction of External Temperature by Quadratic Local Trend Model (0.8) using 6h Data

The prediction intervals are larger than those in linear local trend model.

4 Question 4.1: Seasonal Local Trend Model of Solar Irradiation

As analyzed in section.1, it's reasonable to model the solar irradiation using seasonal local trend model, with reference to page 60 in book.[1]. The fluctuations seems to be larger when the series get bigger, and the general trend is increasing. So multiplicative decomposition is used.

4.1 Trend Component of Solar Irradiation

Trend Component in solar irradiation is extracted using function in library "forecast". Reference to web page for how to use the library Extracting Seasonality and Trend from Data: Decomposition Using R and web page for source Package 'forecast' - R Project. The fig.10 only shows the original trend component, while the model is visualized in fig.12.

```r
library(forecast)
ts_trend_iSolar = ma(dat.f_6h$iSolar[1: n_6h], order = 4, centre = T)
```

Code 4. R Library for Seasonal Local Trend Model and Function to Extract Trend Component

The result is:

$$\hat{\theta} = (87.61334024.95416530.2214459)^T \tag{30}$$

$$\hat{\sigma} = 6.469872 \tag{31}$$

$$V\left[\hat{\theta}\right] = \begin{pmatrix} 21.3357872 & 5.1039112 & 0.41773505 \\ 5.1039112 & 2.3055956 & 0.23124813 \\ 0.4177351 & 0.2312481 & 0.02853357 \end{pmatrix} \tag{32}$$

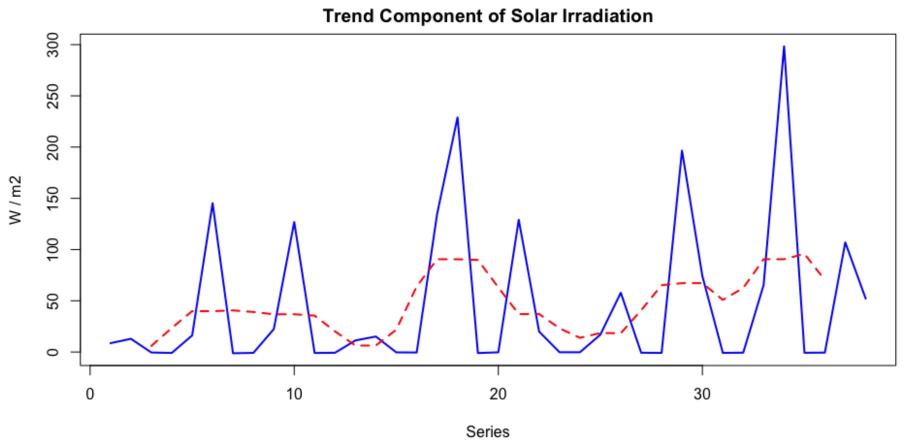

Figure 10. Trend Component of Solar Irradiation using 6h Data

4.2 Averaged Seasonal Component of Solar Irradiation

```r
ts_detrend_iSolar = dat.f_6h$iSolar[1: n_6h] / ts_trend_iSolar
mat_iSolar = t(matrix(data = ts_detrend_iSolar, nrow = 4))
vec_season_iSolar = colMeans(mat_iSolar, na.rm = T)
```

Code 5. R Functions and Operations to Calculate Averaged Seasonal Component

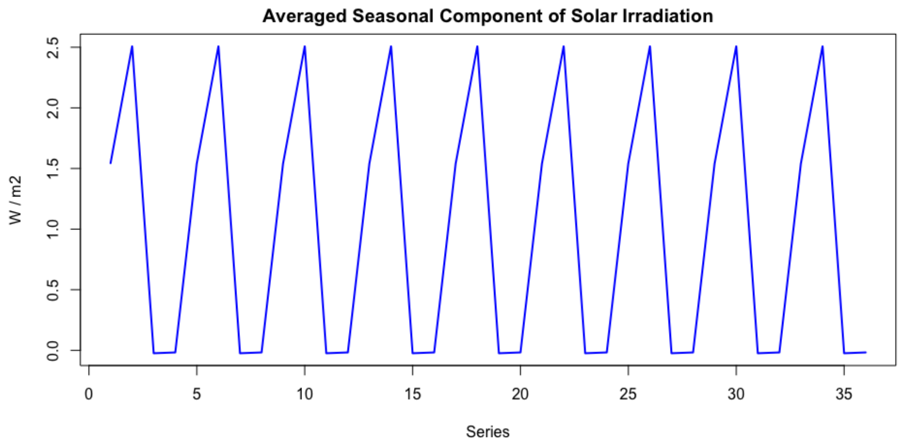

Figure 11. Averaged Seasonal Component of Solar Irradiation using 6h Data

4.3 Prediction of the Solar Irradiation in the Last Day

By predicting the future trend component and multiplying it with the corresponding averaged seasonal component, we can get the future values of seasonal model.

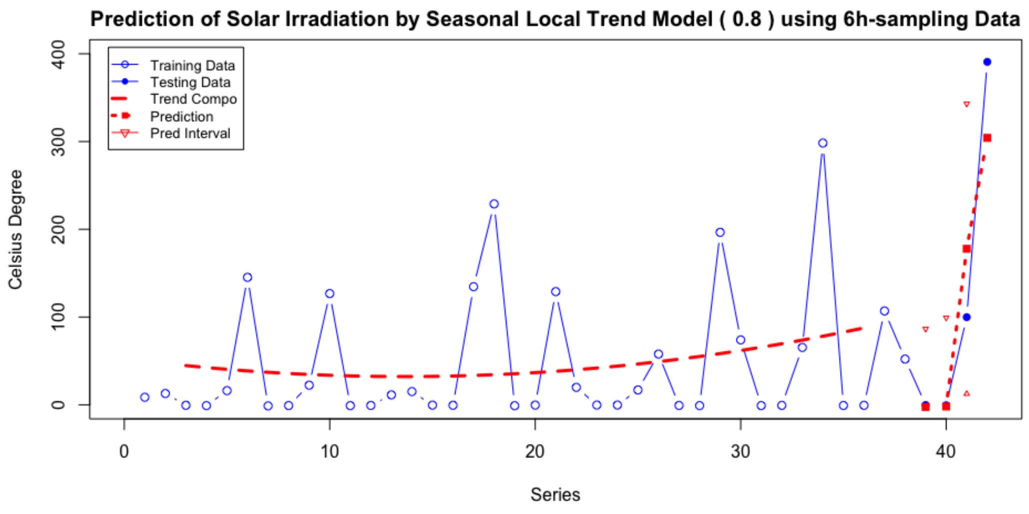

Figure 12. Prediction of Solar Irradiation by Seasonal Local Trend Model (0.8) using 6h-sampling Data

5 Question 4.2: Prediction of the Spatial Heating in the Last Day

Predict the observations for the last day (test data) of the spatial heating based on the 6h data. For this, use the estimated general linear model in question 2, seasonal quadratic local trend model of solar irradiation from question 4.1 and the quadratic local trend model of outdoor temperature from question 3.

$$\hat{\mathbf{T}}_{\text{ex}} = (\hat{T}_{\text{ex}N+1|N}, \hat{T}_{\text{ex}N+2|N}, \hat{T}_{\text{ex}N+3|N}, \hat{T}_{\text{ex}N+4|N})^T \tag{33}$$

$$\hat{\mathbf{I}}_{\text{sol}} = (\hat{I}_{\text{sol}N+1|N}, \hat{I}_{\text{sol}N+2|N}, \hat{I}_{\text{sol}N+3|N}, \hat{I}_{\text{sol}N+4|N})^T \tag{34}$$

$$\hat{\Phi}_{\text{h}} = \hat{H}_{\text{tot}}\hat{T}_{\text{in}} - \hat{H}_{\text{tot}}\hat{T}_{\text{ex}} + \hat{g}\hat{A}_{\text{sol}}\hat{I}_{\text{sol}} \tag{35}$$

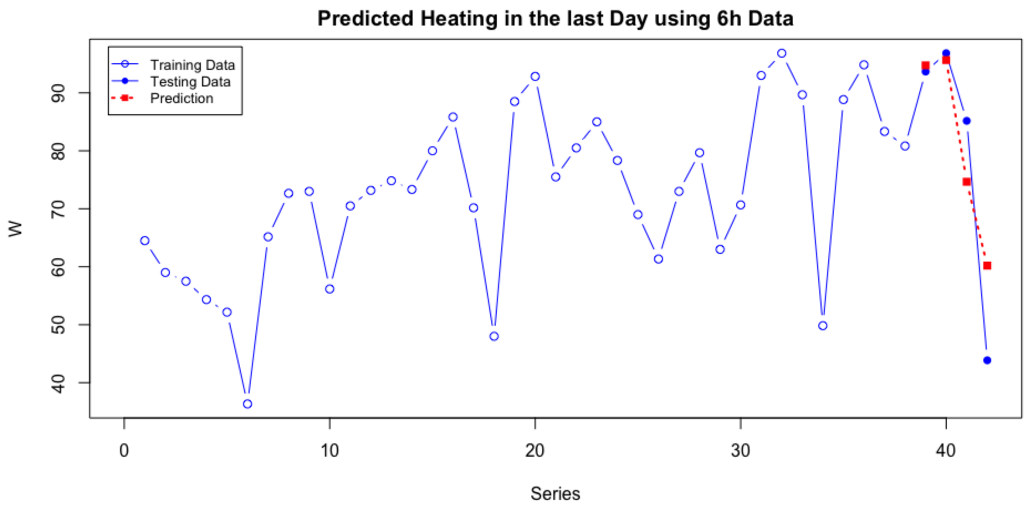

Figure 13. Predicted Heating in the last Day using 6h Data

5.1 Propagation of Uncertainty in Prediction

In statistics, propagation of uncertainty (or propagation of error) is the effect of variables' uncertainties (or errors, more specifically random errors) on the uncertainty of a function based on them.

6 Appendix: Code and Reference

```r
cal_vec_sse.hat <- function(mat_x, vec_y, vec_theta.hat, mat_capSigma = diag(length(mat_x[,1]))){
    # Calculate the sum of squared error (sse), eq.
    vec_epsilon <- vec_y - mat_x %*% vec_theta.hat
    sse.hat <- t(vec_epsilon) %*% solve(mat_capSigma) %*% vec_epsilon
    return(sse.hat)
}
cal_sigma.hat <- function(mat_x, vec_y, vec_theta.hat, mat_capSigma = diag(length(mat_x[,1]))){
    # Estimator for the variance, eq 3.44, Theorem 3.4, P39
    sigma.hat.square <- cal_vec_sse.hat(mat_x, vec_y, vec_theta.hat, mat_capSigma) /
                        (length(mat_x[,1]) - length(vec_theta.hat))
    sigma.hat <- sqrt(drop(sigma.hat.square))
    return(sigma.hat)
}
cal_mat_var.theta.hat <- function(mat_x, sigma.hat, mat_capSigma= diag(length(mat_x[,1]))){
    # Calculate the variance of vec_theta.hat, eq 3.43, P39
    mat_var.theta.hat <- sigma.hat^2 * solve(t(mat_x) %*% solve(mat_capSigma) %*% mat_x)
    return(drop(mat_var.theta.hat))
}
cal_mat_intervalConf <- function(prob = 0.95, mat_x, vec_theta.hat, vec_y){
    n <- length(mat_x[,1])
    p <- length(vec_theta.hat)
    quantileStudentDist <- qt(p = prob, df = n - p)
    vec_var.y.hat <- (vec_y - mat_x %*% vec_theta.hat)^2
    vec_boundUp <- mat_x %*% vec_theta.hat + quantileStudentDist * sqrt(vec_var.y.hat / n)
    vec_boundLow <- mat_x %*% vec_theta.hat - quantileStudentDist * sqrt(vec_var.y.hat / n)
    return(list(vec_boundUp = vec_boundUp, vec_boundLow = vec_boundLow))
}
```

Code 6. Functions for General Linear Model (GLM)

```r
n_6h <- length(dat.f_6h$series) - 4
mat_x_6h <- cbind(1, dat.f_6h$tempExternal[1:n_6h], dat.f_6h$iSolar[1:n_6h])
vec_y_6h <- dat.f_6h$heating[1:n_6h]
vec_theta.hat_ols_6h <- pred_vec_theta.hat(mat_x = mat_x_6h, vec_y = vec_y_6h)
sigma.hat_ols_6h <- cal_sigma.hat(mat_x = mat_x_6h, vec_theta.hat = vec_theta.hat_ols_6h, vec_y = vec_y_6h)
mat_var.theta.hat_ols_6h <- cal_mat_var.theta.hat(mat_x = mat_x_6h, sigma = sigma.hat_ols_6h)
(tempInternal_ols_6h <- vec_theta.hat_ols_6h[1] / - vec_theta.hat_ols_6h[2])
```

Code 7. Estimation of General Linear Model (GLM) using OLS

```r
(rho_expDecay_6h <- cal_rho_expDecayRelaxAlgo(mat_x = mat_x_6h, vec_y = vec_y_6h))
mat_capSigma_expDecay_6h <- do_mat_capSigma_expDecay(rho = rho_expDecay_6h,
                                                                 n = n_6h)
vec_theta.hat_expDecay_6h <- pred_vec_theta.hat(mat_x = mat_x_6h, vec_y = vec_y_6h,
                                        mat_capSigma = mat_capSigma_expDecay_6h)
sigma.hat_expDecay_6h <- cal_sigma.hat(mat_x = mat_x_6h, vec_theta.hat = vec_theta.hat_expDecay_6h, vec_y = vec_y_6h)
mat_var.theta.hat_expDecay_6h <- cal_mat_var.theta.hat(mat_x = mat_x_6h, sigma = sigma.hat_expDecay_6h)
(tempInternal_expDecay_6h <- vec_theta.hat_expDecay_6h[1] / - vec_theta.hat_expDecay_6h[2])
```

Code 8. WLS using 6h Data when Residiuals Have Exponential Decaying Function

```r
do_seq_zeroDecreaseRev <- function(n){return(rev(- seq(0, n-1)))}
do_mat_capSigma_trend <- function(lambda = 1, n){
    mat_capSigma_trend <- diag(n)
    seq_zdi <- do_seq_zeroDecreaseRev(n)
    for (i in 1: n){
        mat_capSigma_trend[i, i] <- 1 * lambda^seq_zdi[i]
    }
    return(mat_capSigma_trend)
}
cal_mat_capF_trend <- function(mat_x_trend, mat_capSigma_trend){
    mat_capF <- t(mat_x_trend) %*% solve(mat_capSigma_trend) %*% mat_x_trend
    return(mat_capF)
}
cal_mat_h_trend <- function(mat_x_trend, mat_capSigma_trend, vec_y_trend){
    mat_h <- t(mat_x_trend) %*% solve(mat_capSigma_trend) %*% vec_y_trend
    return(mat_h)
}
cal_vec_theta.hat_trend <- function(mat_x_trend, mat_capSigma_trend, vec_y_trend){
    mat_capF <- cal_mat_capF_trend(mat_x_trend, mat_capSigma_trend)
    mat_h <- cal_mat_h_trend(mat_x_trend, mat_capSigma_trend, vec_y_trend)
    vec_theta.hat_trend <- solve(mat_capF) %*% mat_h
    return(vec_theta.hat_trend)
}
cal_vec_intervalPred <- function(prob = 0.95, n, p, y.hat, var){
    quantileStudentDist <- qt(p = 0.95, df = n - p)
    boundUp <- y.hat + quantileStudentDist * sqrt(var)
    boundLow <- y.hat - quantileStudentDist * sqrt(var)
    return(list(boundUp = drop(boundUp), boundLow = drop(boundLow)))
}
cal_mat_intervalPred <- function(vec_l, prob = 0.95, n, p, vec_y.hat, vec_var){
    vec_boundUp <- numeric(length(vec_l))
    vec_boundLow <- numeric(length(vec_l))
    for (l in vec_l){
        interval <- cal_vec_intervalPred(prob, n, p, y.hat = vec_y.hat[l], var = vec_var[l])
        vec_boundUp[l] <- interval$boundUp
        vec_boundLow[l] <- interval$boundLow
    }
    return(list(vec_boundUp = vec_boundUp, vec_boundLow = vec_boundLow))
}
cal_vec_memoryTotal <- function(lambda, n){
    vec_memoryTotal <- numeric(n)
    vec_memoryTotal[1] <- 1
    for (j in 1: (n-1)){
     vec_memoryTotal[j+1] <- vec_memoryTotal[j] + lambda^j
    }
    return(vec_memoryTotal)
}
```

Code 9. Functions for global and local trend model

```
func_f_linear <- function(j){
    return(rbind(1, j))
}
pred_y_trend_linear <- function(l, vec_theta.hat_trend){
    y_pred_trend <- t(func_f_linear(l)) %*% vec_theta.hat_trend
    return(y_pred_trend)
}
pred_var_trend_linear <- function(l, mat_x_trend, vec_y_trend, vec_theta.hat_trend, mat_capSigma_trend){
    sigma.hat <- cal_sigma.hat(mat_x = mat_x_trend, vec_y = vec_y_trend,
                               vec_theta.hat = vec_theta.hat_trend, mat_capSigma = mat_capSigma_trend)
    mat_capF_trend_trend <- cal_mat_capF_trend(mat_x_trend, mat_capSigma_trend)
    var_pred_trend <- sigma.hat^2 %*% (1 + t(func_f_linear(l)) %*% solve(mat_capF_trend_trend) %*% func_f_linear(l))
    return(var_pred_trend)
}
pred_vec_y_trend_linear <- function(vec_l, vec_theta.hat_trend){
    vec_y_trend_linear <- numeric(length(vec_l))
    for (i in (1: length(vec_l))){
        vec_y_trend_linear[i] <- pred_y_trend_linear(vec_l[i], vec_theta.hat_trend)
    }
    return(vec_y_trend_linear)
}
pred_vec_var_trend_linear <- function(vec_l, mat_x_trend, vec_y.pred_trend, vec_theta.hat_trend, mat_capSigma_trend){
    vec_var_trend_linear <- numeric(length(vec_l))
    for (i in (1: length(vec_l))){
        vec_var_trend_linear[i] <- pred_var_trend_linear(vec_l[i], mat_x_trend,
                                                         vec_y.pred_trend[i], vec_theta.hat_trend, mat_capSigma_trend)
    }
```

```r
    return(vec_var_trend_linear)
}
```

Code 10. Functions for Linear Global/Local Trend Model

```r
func_f_quadratic <- function(j){
    return(rbind(1, j, j^2 / 2))
}
pred_y_trend_quadratic <- function(l, vec_theta.hat_trend){
    y_pred_trend <- t(func_f_quadratic(l)) %*% vec_theta.hat_trend
    return(y_pred_trend)
}
pred_vec_y_trend_quadratic <- function(vec_l, vec_theta.hat_trend){
    vec_y.pred_trend_quadratic <- numeric(length(vec_l))
    for (i in seq(1, length(vec_l))){
        vec_y.pred_trend_quadratic[i] <- pred_y_trend_quadratic(vec_l[i], vec_theta.hat_trend)
    }
    return(vec_y.pred_trend_quadratic)
}
pred_var_trend_quadratic <- function(l, mat_x_trend, vec_y_trend, vec_theta.hat_trend, mat_capSigma_trend){
    sigma.hat <- cal_sigma.hat(mat_x = mat_x_trend, vec_y = vec_y_trend,
                              vec_theta.hat = vec_theta.hat_trend, mat_capSigma = mat_capSigma_trend)
    mat_capF_trend_trend <- cal_mat_capF_trend(mat_x_trend, mat_capSigma_trend)
    var.pred_trend <- sigma.hat^2 %*% (1 + t(func_f_quadratic(l)) %*% solve(mat_capF_trend_trend) %*% func_f_quadratic(l))
    return(var.pred_trend)
}
pred_vec_var_trend_quadratic <- function(vec_l, mat_x_trend, vec_y.pred_trend, vec_theta.hat_trend, mat_capSigma_trend){
    vec_var.pred_trend_quadratic <- numeric(length(vec_l))
    for (i in seq(1, length(vec_l))){
        vec_var.pred_trend_quadratic[i] <- pred_var_trend_quadratic(vec_l[i],
                                                                    mat_x_trend,
                                                                    vec_y.pred_trend[i],
                                                                    vec_theta.hat_trend,
                                                                    mat_capSigma_trend)
    }
    return(vec_var.pred_trend_quadratic)
}
```

Code 11. R Functions for Quadratic Global/Local Trend Model

References

1. Madsen, H. *Time series analysis* (Chapman and Hall/CRC, 2007).

2. Goodwin, G. C. & Payne, R. L. *Dynamic system identification: experiment design and data analysis*, vol. 136 (Academic press New York, 1977).