Technical University of Denmark

Time Series Analysis

02417

Assignment 1

Authors: Jiayi Han (s164229)

October 3, 2019

Introduction

The purpose of this assignment is to estimate different linear models for the development of atmospheric CO2 over the past 60 years. In this assignment computations for OLS, WLS and Local trend model will be performed using RStudios.

1 Plotting

This plot contains data about the atmospheric CO_2 (in *ppm*) over time. This plot shows that the CO_2 overall concentration is increasing over the years. The plot will not be shown here, but in exercise OLS and WLS.

2 OLS and WLS

Given follwing model, each model parameter can be estimated:

$$Y_t = \alpha + \beta_t t + \beta_s \sin\left(\frac{2\pi}{p}t\right) + \beta_c \cos\left(\frac{2\pi}{p}t\right) + \varepsilon_t \tag{1}$$

Can be written as following vector - expression, our interest is on finding the 4 parameters in the model. Here we define the period as $p = 12$ months (1 year)

$$Y_t = \begin{pmatrix} 1 & t & \sin\left(\frac{2\pi}{p}t\right) & \cos\left(\frac{2\pi}{p}t\right) \end{pmatrix} \begin{pmatrix} \alpha \\ \beta_t \\ \beta s \\ \beta c \end{pmatrix} + \varepsilon_t \tag{2}$$

The parameters are:

$$\hat{\boldsymbol{\theta}} = \begin{pmatrix} \alpha \\ \beta_t \\ \beta s \\ \beta c \end{pmatrix} = \begin{pmatrix} -2649.7599 \\ 1.5518 \\ -27.5799 \\ 81.7215 \end{pmatrix} \tag{3}$$

The estimation of variance is given by:

$$\hat{\sigma^2} = \frac{S(\hat{\boldsymbol{\theta}})}{n-p} = 4.02^2 \tag{4}$$

where n= 718 (number of observations) and p=4 (number of parameters) The estimated prediction error variance is:

$$V[\hat{\boldsymbol{\theta}}] = \hat{\sigma^2}(\boldsymbol{x}^T\boldsymbol{x}) \tag{5}$$

And the uncertainty of each parameter is then:

$$\sigma = \sqrt{diag(V[\hat{\boldsymbol{\theta}})} = \begin{bmatrix} 17.4869 \\ 0.0088 \\ 0.2129 \\ 0.2141 \end{bmatrix} \tag{6}$$

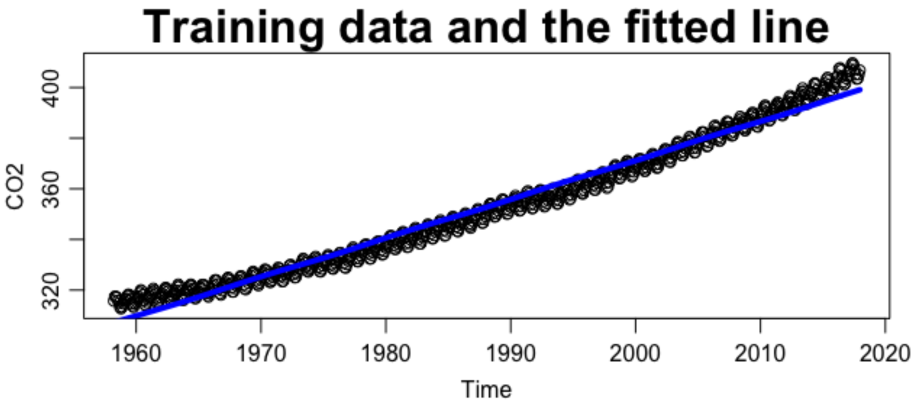

Figure 1: The fitted values together with the training data. The blue line shows the fitted data points based on the parameters in equation 3.

The following equation describes the correlation between the residuals:

$$\operatorname{Cor}\left(\varepsilon_{t_i}, \varepsilon_{t_j}\right) = \rho^{|t_i - t_j|} \quad \text{for some} \quad 0 < \rho < 1 \tag{7}$$

Assuming that the correlation is exponential decaying function of the time distance between two observations. The correlation matrix is then given by:

$$\Sigma = \begin{pmatrix} 1 & \rho & \rho^2 & \cdots \\ \rho & 1 & \rho & \cdots \\ \rho^2 & \rho & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{8}$$

The first iteration is already performed, the matrix Σ (according to page 41) could e.g start out with the identity matrix. (which corresponds to the unweighted case just as the one performed previously in this exercise) The correlation between the residuals can be calculated during each iteration in order to determine the value of ρ.

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{x}\right)^{-1} \boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{Y} \tag{9}$$

The following table shows the values of ρ from five iterations in the relaxation algorithm. It can be seen that the convergence of correlation $[\rho]$ is around 2nd and 3rd iteration.

$$\begin{bmatrix} 0.9529355043 & 0.9532866206 & 0.9532871947 & 0.9532871956 & 0.9532871956 \end{bmatrix}$$

Following table shows the estimated parameters based on each iterations in the relaxation algorithm:

$$\begin{bmatrix} & 1 & 2 & 3 & 4 & 5 \\ \boldsymbol{\alpha} & -2706.91846 & -2701.89285 & -2701.85797 & -2701.85791 & -2701.85791 \\ \boldsymbol{\beta_t} & 1.53910 & 1.53678 & 1.53677 & 1.53677 & 1.53677 \\ \boldsymbol{\beta_s} & 0.16310 & 0.56581 & 0.56603 & 0.56603 & 0.56603 \\ \boldsymbol{\beta_c} & 0.09839 & 0.34100 & 0.34136 & 0.34136 & 0.34136 \end{bmatrix}$$

And the uncertainty of the estimate:

$$V[\hat{\boldsymbol{\theta}}] = \hat{\sigma^2}(\boldsymbol{x}^T\boldsymbol{x})$$

Where σ^2 illustrated for each iteration

$$\begin{bmatrix} 4.020635^2 & 4.019384^2 & 4.034021^2 & 4.034045^2 & 4.034045^2 \end{bmatrix} \tag{10}$$

$$V[\hat{\boldsymbol{\theta}}] = \hat{\sigma^2} \begin{bmatrix} 18.916269 & -0.009514 & -0.016759 & 0.032211 \\ -0.009514 & 0.000005 & 0.000008 & -0.000016 \\ -0.016759 & 0.000008 & 0.002805 & -0.000022 \\ 0.032211 & -0.000016 & -0.000022 & 0.002836 \end{bmatrix} \tag{11}$$

Since the variances are in the diagonal, we know that for each parameter we have an uncertainty of:

$$\sigma = \sqrt{diag(V[\hat{\boldsymbol{\theta}})} = \begin{bmatrix} 17.5452 \\ 0.0088 \\ 0.2136 \\ 0.2148 \end{bmatrix} \tag{12}$$

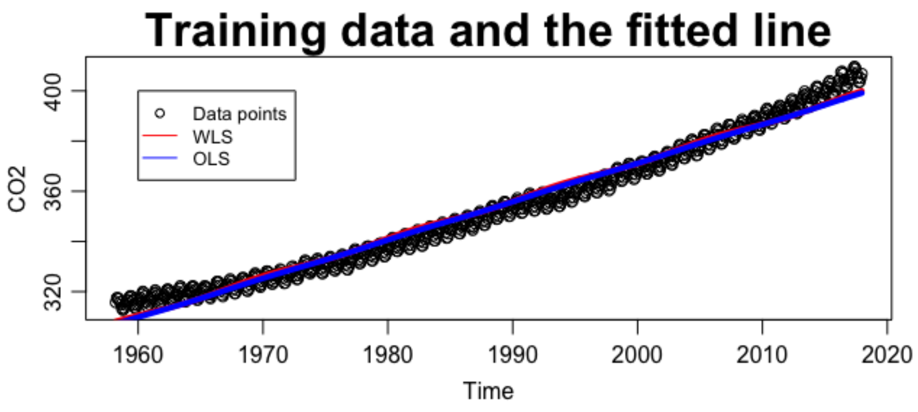

Figure 2: Shows the comparison between the OLS and WLS

There is not a big difference from the WLS and OLS (shown in plot 2)

3 Local linear trend

A trend model is defined as:

$$Y_{N+j} = \boldsymbol{f}^T(j)\boldsymbol{\theta} + \epsilon_{N+j} \tag{13}$$

Where a condition to forecast function $\boldsymbol{f}(j)$ is:

$$f(j+1) = \boldsymbol{L}\boldsymbol{f}(j) \tag{14}$$

Given that we know the linear model shown in equation (1) is a combination of a linear trend model with harmonic trend model then the matrix L (transition matrix) is:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & cos(\frac{2\pi}{p}) & sin(\frac{2\pi}{p}) \\ 0 & 0 & -sin(\frac{2\pi}{p}) & cos(\frac{2\pi}{p}) \end{pmatrix} \tag{15}$$

And the $\boldsymbol{f(j)}$ is:

$$\boldsymbol{f(j)} = \begin{pmatrix} 1 \\ j \\ sin(\frac{2\pi}{p}j) \\ cos(\frac{2\pi}{p}j) \end{pmatrix} \tag{16}$$

Where $\boldsymbol{f(0)}$ is:

$$\boldsymbol{f(0)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{17}$$

By introducing the forgetting factor, the global trend model becomes to local trend model, where it's about forgetting the past observations with exponential smoothing. In this exercise, the forgetting factor is set to $\lambda = 0.9$.

By using the expressed $\boldsymbol{f}(j)$ we will be able to obtain the F_N and h_N as a part of the trend model:

$$F_N = \sum_{j=0}^{N-1} \lambda^j \boldsymbol{f}(-j)^T \boldsymbol{f}(-j) \tag{18}$$

And

$$h_N = \sum_{j=0}^{N-1} \lambda^j \boldsymbol{f}(-j)^T Y_{N-j} \tag{19}$$

By using the F_N and h_N we will be able to obtain the updated estimation of parameters.

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{N}} = \boldsymbol{F}_N^{-1}\boldsymbol{h}_N \tag{20}$$

We wish to have a more accurate prediction of the current value in interest, therefore excluding the first 10 observations, for building the trend model. This means that the first predicted value is $\hat{Y}_{11|10}$.

Using the estimated parameters, the one step prediction can then be obtained by:

$$\hat{Y}_{N+l|N} = \boldsymbol{f}^T(l)\hat{\boldsymbol{\theta}}_{\boldsymbol{N}} \tag{21}$$

From each new observation we get, we can update our model, for the purpose of update the parameters to predict next $\hat{Y}_{N+l|N}$.

$$\boldsymbol{F}_{N+1} = \boldsymbol{F}_{\boldsymbol{N}} + \lambda^N \boldsymbol{f}(-j)^T \boldsymbol{f}(-j) \tag{22}$$

$$\boldsymbol{h}_{N+1} = \lambda \boldsymbol{L}^{-1}\boldsymbol{h}_N + \boldsymbol{f}(0)Y_{N+1} \tag{23}$$

Then updating the estimated parameters:

$$\hat{\boldsymbol{\theta}}_{N+1} = F_{N+1}^{-1} h_{N+1} \tag{24}$$

Updates will be performed for each new observation, therefore the "N" is a increasing number.

The one step prediction is presented in figure 3 together with the confidence interval of 95%.

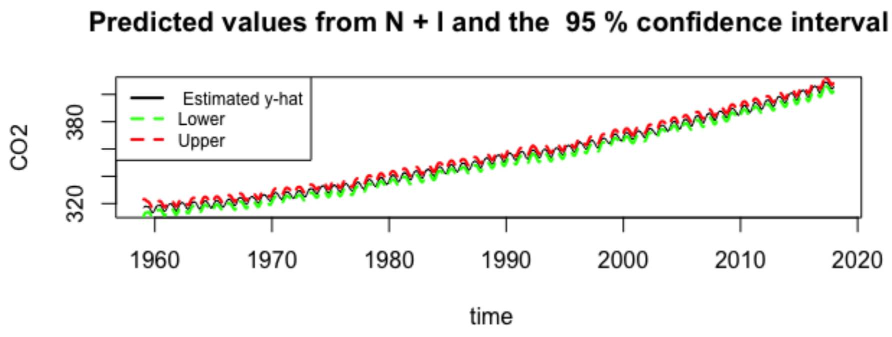

Figure 3: The predicted values of $\hat{Y}$ by one-step prediction.

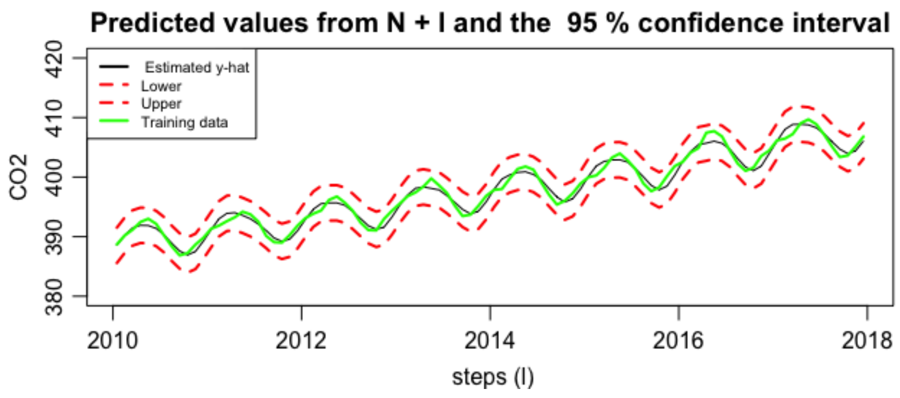

Figure 4: This is the zoomed version of one-step prediction together with the confidence interval of 95%

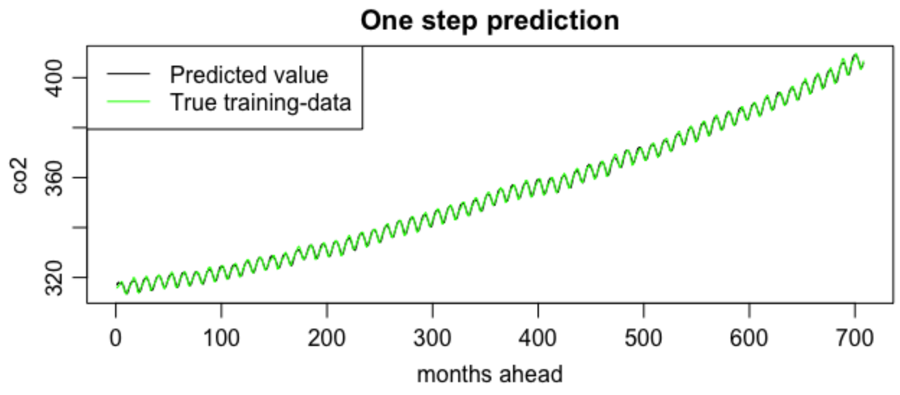

Figure 5: The predicted value and the actual data.

It is very clear from figure 5, that the one step prediction values are relatively close to the true data.

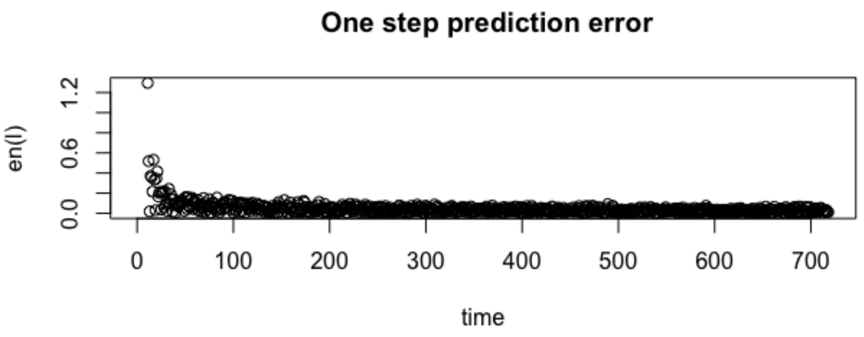

Figure 6: The one step prediction error. The value of error converges towards 2.024111e-05. This plot is an illustration of the absolute value of the prediction error: $|Y_N - \hat{Y}_N|$

And the estimated σ for each predicted observation is calculated based on following expression:

$$\hat{\sigma}^2 = \left(\boldsymbol{Y} - \boldsymbol{x}_N\widehat{\boldsymbol{\theta}}_N\right)^T \Sigma^{-1} \left(\boldsymbol{Y} - \boldsymbol{x}_N\widehat{\boldsymbol{\theta}}_N\right) / (T-p), \quad T > p \tag{25}$$

Computations of σ is performed for each prediction. Where the T is:

$$T = \sum_{j=0}^{N-1} \lambda^j = \frac{1-\lambda^N}{1-\lambda} \tag{26}$$

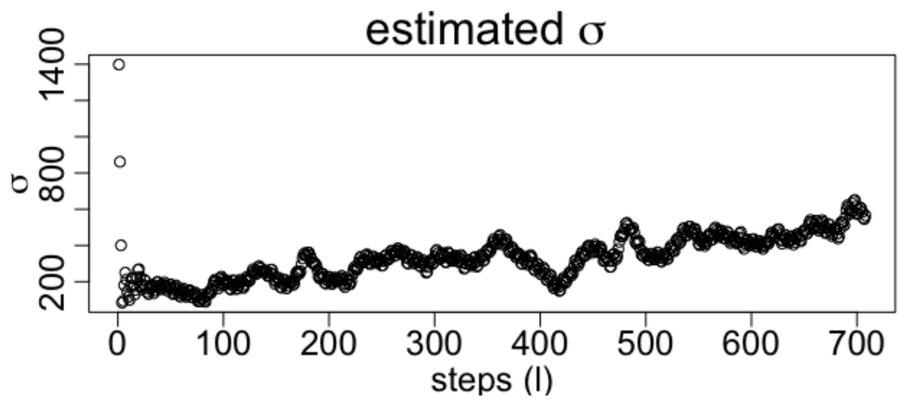

Figure 7: This is the estimated σ, that is calculated from equation 25.

From this plot in figure 7 it is clear that the σ is large in the beginning. The minimum standard deviation is 0, and mean value of the standard deviation is

328.97.

Further predictions with 1, 2, 6, 12 and 20 months

This section we will utilize the same algorithm as before. Here we will predict with a larger step without updating the parameters, as this is a simulation of future prediction without any given data values to update our trend model. Following table shows the values that were predicted:

$$\begin{bmatrix} 408.21227 \\ 410.07926 \\ 410.83243 \\ 408.99884 \\ 410.64367 \end{bmatrix} \tag{27}$$

The values can be compared with the test data that were previously excluded from the training data: Prediction errors future $\epsilon = |Y - \hat{Y}|$:

$$\begin{bmatrix} 0.25227 \\ 1.75926 \\ 0.04243 \\ 0.07116 \\ 0.69367 \end{bmatrix} \tag{28}$$

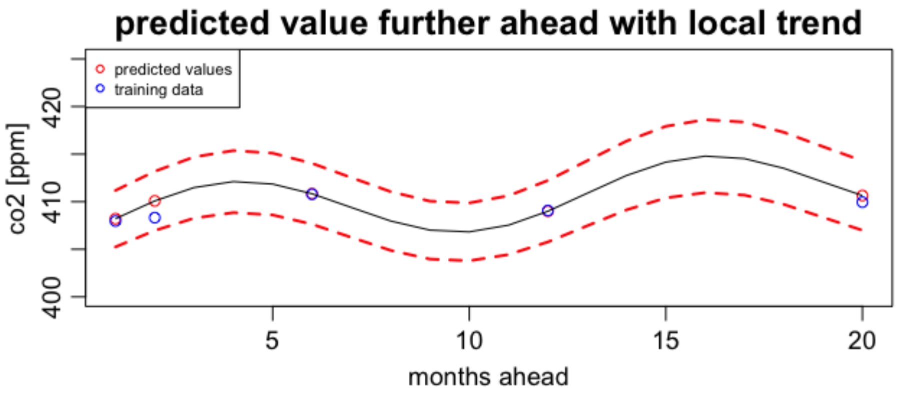

Figure 8: This plot shows only some selected values of interest in order to compare the true values together with the predicted values. The red dots are representing the $\hat{Y}_{MonthsAhead}$, whereas the blue dots are the true value extracted from the test data. Red lines represents the

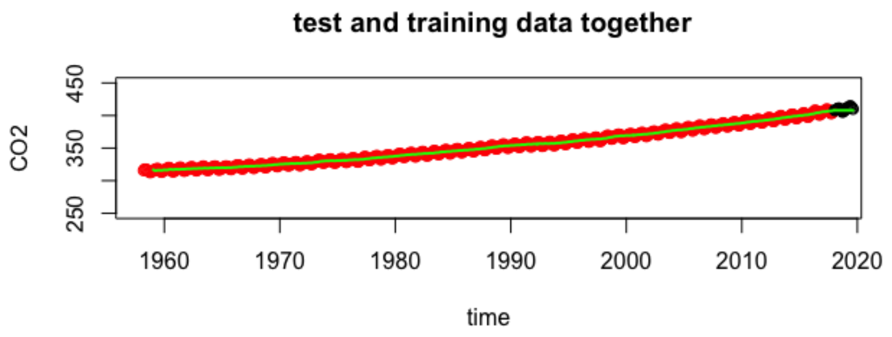

Figure 9: The data and the estimated mean for each time step

From the assumption about that the estimated mean is the first element in θ_t from each time point, by this the values can be extracted. It can be seen that during the "red" part of the data, where the estimated parameters are updated (figure 9) we can see an increasing mean value. For the test data, we consider this part as unobserved points, and the mean is starting to deviate from the test-data as we predict further ahead from the last observation.

4 Optimal λ

An optimal value of λ can be selected by minimizing the squared error

$$S(\alpha) = \sum_{t=1}^{N} e_t^2(\alpha) = \sum_{t=1}^{N} (Y_t - \hat{Y}_{t|t-1}(\alpha))^2 \tag{29}$$

From that expression, we are sure that the outcome of the sum of squared one step prediction errors is dependent on the chosen value of λ, which is defined by $\alpha = 1 - \lambda$. By this, the purpose of this section is to calculate and evaluate $S(\alpha)$, where the first observations are excluded (burning period is 100 months).

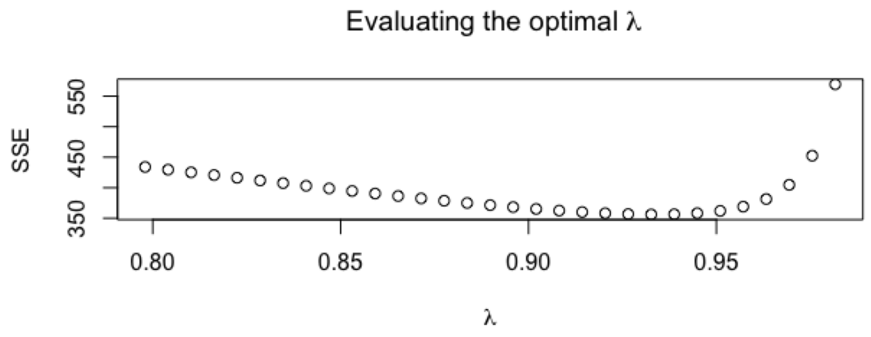

Figure 10: When looking at the optimal lambda, which is the one that gives the lowest SSE, at this case $\lambda_{optimal} = 0.9326531$

Based on the selected lambda (forgetting factor), the one step prediction will be re-calculated for the period 2010 and further ahead:

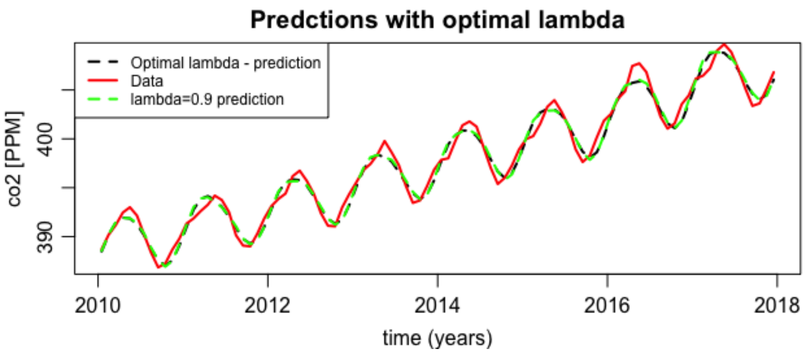

Figure 11: Comparing the prediction. Since there is not a big difference between the provided $\lambda = 0.9$ and optimal $\lambda_{optimal} = 0.933$, there is not a big prediction difference shown in this plot.

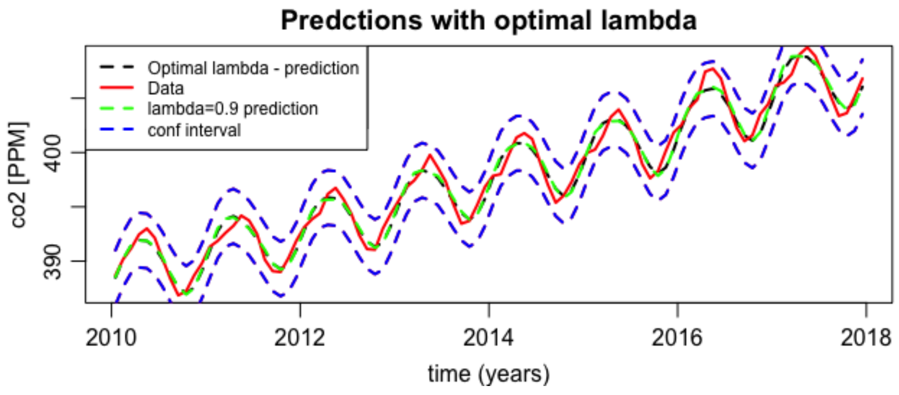

Figure 12: Here is a plot with added confidence intervals of 95%

Following table shows the predictions of 1, 2, 6, 12, 20 months:

$$\begin{bmatrix} 408.07386 \\ 409.91014 \\ 410.74633 \\ 408.84418 \\ 410.55102 \end{bmatrix} \tag{30}$$

And we can calculate the prediction errors as before ($|Y - \hat{Y}|$):

$$\begin{bmatrix} 0.11386 \\ 1.59014 \\ 0.04367 \\ 0.22582 \\ 0.60102 \end{bmatrix} \tag{31}$$

In comparison with the previous λ value, the prediction erros is slightly less. Both in (31) and (28) has a larger error around 2.months prediction compared to the other months.

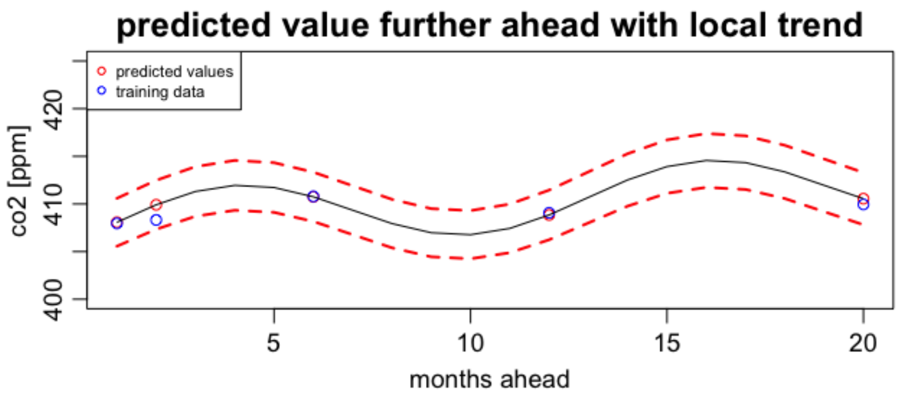

Figure 13: Predictions further ahead using the new value of λ

5 Over all

The overall comparison of the models: Both the OLS and WLS were easy accessible models, but the thing is that they failed to model the sinusoid part of the actual data (oscillations) as they were too simple. The local trend model were performing very well in terms of modelling the oscillating atmosphere CO_2 concentrations.
During the training trend model, the overall mean was increasing, which is a good representation of the actual CO_2 concentrations. But when predicting ahead, as shown in 9, the mean value would be considered as constant which would lead to a poor prediction.
At this part, we can see that the OLS WLS is performing better in terms of representing the overall increasing CO_2 concentrations.
It depends on what the purpose of modelling is, the local trend model will be able to predict quite accurate within small distances from the last observation. If a rough estimate of future CO_2 concentrations (approx 20 years), based on the assumption: that the mean-concentration will change, I would prefer the WLS model.

```
#######################################################################################
# Jiayi Han
# s164229
# 2019 F Time Series Analysis (02417)

##############Definition of function for later use ##############################

bmatrix = function(x, digits=NULL, ...) {
  library(xtable)
  default_args = list(include.colnames=FALSE, only.contents=TRUE,
                      include.rownames=FALSE, hline.after=NULL, comment=FALSE,
                      print.results=FALSE)
  passed_args = list(...)
  calling_args = c(list(x=xtable(x, digits=digits)),
                   c(passed_args,
                     default_args[setdiff(names(default_args), names(passed_args))]))
  cat("\\begin{bmatrix}\n",
      do.call(print.xtable, calling_args),
      "\\end{bmatrix}\n")
}
##############Definition of function for later use ##############################
# This function generates the Sigma-matrix
Get_Sigma <- function(N, r){
  G <- matrix(0, nrow = N, ncol=N)
  for(i in 1:N)
  {
    for (k in 1:N)
    {
      G[i,k] <-r^(abs(i-k))
    }
  }
  G
}
#######################################################################################
setwd("~/Desktop/7.semester/Introduction to Time Series Analysis/Assignment1")
# Load the txt file
data<- read.table("A1_co2.txt", header=TRUE)
str(data) # To get some info about the data
TrainData<- data[1:718,]
TestData<- data[719:738,]
N=length(TrainData$co2)
p=12
#######################################################################################
# 1.1 Plotting :
par(mfrow=c(1,1))
par(mgp=c(2, 0.7,0), mar=c(3,3,2,1))
plot(TrainData$time, TrainData$co2, xlab='Time', ylab='CO2')
# This data can be fitted with OLS-estimates (Ordinary Least Squares)


############################Manual finding parameter: ########################
X <- cbind(1, TrainData$time, sin(2*pi*TrainData$time/p), cos(2*pi*TrainData$time/p))
# head(X)
(theta <- solve(t(X)%*%X, t(X)%*% TrainData$co2))
# Use the parameters to fit the model
fit_line_manual<-(X) %*% theta
# calculate residual:
residual_manual<-sqrt((t(TrainData$co2-fit_line_manual) %*% (TrainData$co2-
fit_line_manual))/(length(TrainData$co2)-4))
# Degree of freedom is 4 since we are working with the given model...
(residual_manual)
epsilon_manual<- (TrainData$co2-fit_line_manual);
# The variance
sigma.hat <-  sqrt(sum(epsilon_manual^2) / (length(epsilon_manual)-4) )
# the forrelation is:
Correlation_manual  <- cor( epsilon_manual[1:N-1] , epsilon_manual[2:N] )


############################ USE the lm to validate ###############################
xc<-cos(2*pi*TrainData$time/p)
xs<-sin(2*pi*TrainData$time/p)
fit.lm<-lm(co2~time+xs+xc, data=TrainData)
summary(fit.lm)

## Estimate of variance
names(fit.lm)
summary(fit.lm)
sigma.hat <-  sqrt(sum(fit.lm$residuals^2) / (length(TrainData$time)-4) )

##########################  Plot s ##################################################
plot(TrainData$time, TrainData$co2,
    main = "Training data and the fitted line" ,xlab='Time', ylab='CO2',
    cex.main = 2, cex.lab = 1, cex.axis =1 )
# the LM
points(TrainData$time,fit.lm$fitted.values,type='l',col='red', lwd = 4)
# the manual
points(TrainData$time,fit_line_manual,type='l',col='blue', lwd = 4)

################################### Relaxation #####################################
# the correlation between the residuals.
```

```
X <- cbind(1, TrainData$time, sin(2*pi*TrainData$time/p), cos(2*pi*TrainData$time/p))
# head(X)
rho <- 0 # for first ilteration with the identity matrix
N=length(TrainData$time)

theta_ilteration=matrix(0, nrow = 4, ncol=5)
fit_line_ilteration= matrix(0, nrow = N, ncol=5)
residual_ilteration = matrix(0, nrow = N, ncol=5)
epsilon_ilteration = matrix(0, nrow = N, ncol=5)
Correlation_ilteration=matrix(0, nrow = 1, ncol=5)
sigma.hat.ilt=matrix(0, nrow = 1, ncol=5)
for (ilteration_relax in 1:5){

  sigma_matrix=Get_Sigma(N, rho);

  theta_ilteration[,ilteration_relax ] <- solve(t(X)%*% solve(sigma_matrix)%*% X,
t(X)%*%solve(sigma_matrix)%*% TrainData$co2)
  # Use the parameters to fit the model
  fit_line_ilteration[,ilteration_relax ]<-(X) %*% theta_ilteration[,ilteration_relax ]
  # calculate residual:
  residual_ilteration[,ilteration_relax ]<-sqrt((t(TrainData$co2-
fit_line_ilteration[,ilteration_relax ]) %*% (TrainData$co2-fit_line_ilteration[,ilteration_relax ]))
                /(length(TrainData$co2)-4))
  epsilon_ilteration[,ilteration_relax ]<-(TrainData$co2-fit_line_ilteration[,ilteration_relax ]);
  epsilon_rho<-epsilon_ilteration[,ilteration_relax ];
  Correlation_ilteration[ilteration_relax]  <- cor( epsilon_rho[1:N-1] , epsilon_rho[2:N] )
  sigma.hat.ilt[ilteration_relax] <-  sqrt( t(epsilon_rho)%*%
solve(sigma_matrix)%*%epsilon_rho/(length(epsilon_rho)-4) )
  rho= Correlation_ilteration[ilteration_relax]
}
X <- cbind(1, TrainData$time, sin(2*pi*TrainData$time/p), cos(2*pi*TrainData$time/p))
## OLS variance of estimate:
x_mat=solve(t(X)%*%X)
var_estimat<-cbind(diag(sigma.hat^2*x_mat))
bmatrix(sqrt(var_estimat), digits=4)

## WLS variance of estimate:
var_estimat_WLS<-cbind(diag(sigma.hat.ilt[5]^2*x_mat))
bmatrix(sqrt(var_estimat_WLS), digits=4)

# export of figures to latex
bmatrix(Correlation_ilteration, digits=10)
bmatrix(theta_ilteration, digits=5)
bmatrix(x_mat, digits=8)
bmatrix(sigma.hat.ilt, digits=6)

rbind(Correlation_ilteration, sigma.hat.ilt )

#Defining one new X so the plot can extent to the full length of data
X <- cbind(1, TrainData$time, sin(2*pi*TrainData$time/p), cos(2*pi*TrainData$time/p))
fit_line_WLS<-(X) %*% theta_ilteration[,5]
fit_line_manual<-(X) %*% theta
############### Plotting for comparison #################
plot(TrainData$time, TrainData$co2,
    main = "Training data and the fitted line" ,xlab='Time', ylab='CO2',
    cex.main = 2, cex.lab = 1, cex.axis =1 )
points(TestData$time,TestData$co2, pch="l", col='green')
points(TrainData$time,fit_line_WLS,col='red',pch="l")
# the manual
points(TrainData$time,fit_line_manual,col='blue', lwd = 1,pch="l")
legend(1960, 400, legend=c("Data points","WLS","OLS"),
      col=c("black","red", "blue"), pch = c(1,NaN,NaN),
lty=c(NaN,1,1), cex=0.8)

########################### Trainsition matrix ####################################
p=12
L = matrix(  c(1, 0, 0, 0,
            1, 1, 0 ,0 ,
            0 ,0 , cos(2*pi/p), sin(2*pi/p),
            0, 0, -sin(2*pi/p), cos(2*pi/p)),
nrow=4,             # number of rows
ncol=4,             # number of columns
byrow = TRUE)       # fill matrix by rows
(L)
LInv <- solve(L)
# Filter the data with the chosen model -(Smoothing)

f <-function(t) rbind( 1, t, sin(2*pi/p*t), cos(2*pi/p*t))
#FN=t(X)%*%X
#hN=t(X)%*%TrainData$time
# Defining F_N and h_N according to the trend model:
lambda <- 0.9
F <- matrix(0, nrow=4, ncol=4)
h <- matrix(0,nrow=4,ncol=1)
Y <-TrainData$co2
NN <- 10
X <- cbind(1, TrainData$time, sin(2*pi*TrainData$time/p), cos(2*pi*TrainData$time/p))

for (j in 0:(NN-1)){  #  first 10 observations as transient
```

```
  F <- F + lambda^(j)*f(-j)%*%t(f(-j))
  h <- h + lambda^(j)*f(-j)*Y[NN-j]
}
#F_10 <- F
#h_1 <-
#theta_10 <- solve(F_10)%*% h_10
# updating the model with one-step prediction:
# F_N+1=F_N+lamda^N*f(-N)%*%f(-N).
theta_update<-matrix(0, nrow= 4, ncol = 718)
theta_update[,10]<- solve(F)%*% h
pred_scale<- matrix(0, nrow = 1 , ncol=718)
SigmaInv <-diag(lambda^(N-(1:N)))
estimated_y_hat <- matrix(0, nrow= 1, ncol = 718)

S_theta_N <- matrix(0, nrow= 1, ncol = 718)
Sig_local<-matrix(0, nrow= 1, ncol = 718)
interval <- matrix (0, nrow= 2, ncol=718)
SsSig<-matrix(0, nrow= 1, ncol = 718)
SSig<-matrix(0, nrow= 1, ncol = 718)
sigma.scale<- rep(NA, N)
e22<-matrix(0, nrow= 1, ncol = 718)
for (j in (NN+1):(N)) # start from N+1 = 11
{
  sigma.scale[j] <- 1 + t(f(1)) %*% solve(F) %*% f(1)
  F <- F + lambda^(j-1)*f(-(j-1))%*%t(f(-(j-1)))
  h <- lambda*(LInv%*%h)+ f(0)%*%Y[j]
  theta_update[,j] <- solve(F)%*% h
  pred_scale[j]<- 1 + t(f(1))%*%solve(F)%*%f(1)

  #S_theta_N[j] <- t(estimated_y_hat[1:(j)]- data$co2[1:(j)]) %*% SigmaInv[(718-j+1):718,(718-
j+1):718] %*% (estimated_y_hat[1:(j)]- data$co2[1:(j)])
  # start from N+1 = 11
  # theta starts from 10 and the estimated y hat from 11
  estimated_y_hat[j]<- t(f(1))%*% theta_update[,j-1]

  interval[,j] <- estimated_y_hat[j] + c(-1,1) * qt(0.975,T-4)* sqrt(pred_scale[j])
  # forcing hthe S(theta,N) to start from 11
  Tjek <- t(f(1))%*% theta_update[,j];
  temp <- matrix(0, nrow = 1, ncol=1)
  for(jj in 1:j)
  {
    temp <- temp+lambda^(j-jj)*(data$co2[jj]- Tjek)^2
  }
  S_theta_N[j] <- temp
  T <- (1-lambda^(j))/(1-lambda)
  SSig[j]<- (t(data$co2[1:(j)]- t(X[j,])%*% theta_update[,j-1] ) %*% SigmaInv[(718-j+1):718,(718-
j+1):718] %*% (data$co2[1:(j)]-t(X[j,])%*% theta_update[,j-1]))/(T-4)
  #SsSig[j]<- (t(data$co2[1:(j)]-estimated_y_hat[(j)]) %*% SigmaInv[(718-j+1):718,(718-
j+1):718] %*% (data$co2[1:(j)]-estimated_y_hat[(j)]))/(T-4)
  #S_theta_N[j] <- t(t(f(j-1:j))%*%theta_update[,j-1] - data$co2[1:(j)]) %*% SigmaInv[(718-
j+1):718,(718-j+1):718] %*% (t(f(j-1:j))%*%theta_update[,j-1] - data$co2[1:(j)])
  Sig_local[j] <- S_theta_N[j] /(T-4)

}


####################################################################################
### Calculation of confidence interval :
lower_bond<- interval[1,11:718]
upper_bond<- interval[2,11:718]
####################################################################################
### Plotting of the predictions and the confidence interval
plot(TrainData$time[11:718], estimated_y_hat[11:718], type = "l", lty=1, xlab= " time", ylab=
"CO2"
     ,  main = "Predicted values from N + l and the  95 % confidence interval")
lines(TrainData$time[11:718],lower_bond, type = "l", lty=2, col="green" , lwd = 2)
lines(TrainData$time[11:718],upper_bond, type = "l", lty=2, col = "red",  lwd = 2)
legend("topleft",legend = c(" Estimated y-hat", "Lower", "Upper")
       ,col=c( "black", "green", "red"),lty=c(1,2,2),lwd=2, cex = 0.75)

####################################################################################
### Plotting of the predictions and the confidence interval
### Plotting of the predictions and the confidence interval
plot(TrainData$time[623:718],estimated_y_hat[623:718], type = "l", lty=1, xlab= " steps (l)", ylab=
"CO2"
     ,  main = "Predicted values from N + l and the  95 % confidence interval", ylim=c(380,420))
lower_bond<- interval[1,623:718]
upper_bond<- interval[2,623:718]
lines(TrainData$time[623:718],lower_bond, type = "l", lty=2, col="red" , lwd = 2)
lines(TrainData$time[623:718],upper_bond, type = "l", lty=2, col = "red",  lwd = 2)
lines(TrainData$time[623:718],TrainData$co2[623:718], type = "l", lty=1, col = "green",  lwd = 2)
op <- par(cex = 1)
legend("topleft",legend = c(" Estimated y-hat", "Lower", "Upper", "Training data")
       ,col=c( "black","red","red","green"),lty=c(1,2,2,1), cex = 0.65, lwd=2 )


## Global estimate of \sigma at end:
last <- N
first <- 10
time <- 1:last
```

```
e2 <- (Y[11:718]-estimated_y_hat[11:718])^2
# plot the running sigma
e2s <- e2/sigma.scale
e2s[1:first] <- 0
ce2s <- cumsum(e2s)
for (i in (first+1):length(ce2s)){
  ce2s[i] <- ce2s[i]/(i-first)
}
ce2s[1:first] <- NA
Sig_local[1:first]  <- NA

e2[1:first] <- 0
ce2 <- cumsum(e2)
ce2[1:(first)] <- NA


# plot the estimate over sigma

plot(sqrt(SSig[11:N]),
    main = expression( paste("estimated ", sigma ," ")) ,ylab = expression(sigma) , xlab = "steps (l)",
    cex.main = 2, cex.lab = 1.5, cex.axis =1.5 )
lines(sqrt(ce2s))

lines(time,sqrt( ce2s/(time-first) ),lwd=2,col=2)
lines(time, sqrt( ce2/(time-first ) ), lwd=2, col=5)
legend("topright",
      legend = c("Running mean", "Normed running mean"),
      pch = c(1,NaN),
      lty=c(NaN,1), col=c("black", "red"))

plot(TrainData$time[11:718], sqrt(Sig_local[11:718]) ,
    main = expression( paste(sigma ," over time")) ,ylab = expression(sigma) , xlab = "time",
    cex.main = 2, cex.lab = 1.5, cex.axis =1.5 )
plot(TrainData$time[11:718],sqrt(SSig[11:718]), xlab = "time", ylab = expression(sigma))

###############################################################################
# Plot the one step prediction error
fo <- TrainData$time[11:718]

plot(sqrt(e2/(time-first)) ,
    main = " One step prediction error", xlab=" time ", ylab="en(l)" )
###############################################################################
plot(estimated_y_hat[11:718], type = "l", lty=1,  xlab="months ahead", ylab="co2", main="One
step prediction")
lines(TrainData$co2[11:718],col="green")
```

```
legend("topleft", legend=c("Predicted value", "True training-data")
       ,lty=c(1,1), col=c("black", "green"))

#########################################################################################
#lambda <- 0.9
lambda<-lambda_test_4[which(SSSS==min(SSSS))]

F_new <- matrix(0, nrow=4, ncol=4)
h_new <- matrix(0,nrow=4,ncol=1)
Y <-TrainData$co2
NN <- 10
for (j in 0:(N-1)){  #  first 10 observations as transient
  F_new <- F_new + lambda^(j)*f(-j)%*%t(f(-j))
  h_new<- h_new + lambda^(j)*f(-j)*Y[N-j]
}
theta_new<-solve(F_new)%*%h_new
one_month_pred <- t(f(1))%*%theta_new
two_month_pred <-  t(f(2))%*%theta_new
six_month_pred <-  t(f(6))%*%theta_new
twelve_month_pred <-  t(f(12))%*%theta_new
twenty_month_pred <-  t(f(20))%*%theta_new
ahead <- rbind ( abs( TestData$co2[1]-one_month_pred) ,
       abs( TestData$co2[2]-two_month_pred),
       abs( TestData$co2[6]- six_month_pred),
       abs( TestData$co2[12]- twelve_month_pred),
       abs( TestData$co2[20]- twenty_month_pred))
bmatrix(ahead, digits=5)
aheadn <- rbind ( ( one_month_pred) ,
                ( two_month_pred),
                ( six_month_pred),
                ( twelve_month_pred),
                ( twenty_month_pred))
bmatrix(aheadn, digits=5)
pred_scale_future<-matrix(0, nrow= 1, ncol=20)
intervalll<-matrix(0, nrow= 2, ncol=20)
estimated_y_hat_it<-matrix(0, nrow= 1, ncol=20)
for (itera in 1:20)
{
  pred_scale_future[itera] <- 1 + t(f(itera))%*%solve(F_new)%*%f(itera)
  TTT <- (1-lambda^(N))/(1-lambda)
  estimated_y_hat_it[itera]<-(t(f(itera))%*%theta_new)
  intervalll[,itera] <- (t(f(itera))%*%theta_new)+ c(-1,1)* qt(0.975,TTT-4)*
sqrt(pred_scale_future[itera])
}
```

```
plot(c (1,2,6,12,20),aheadn, col="red", main = "predicted value further ahead with local trend",
    xlab = "months ahead", ylab = "co2 [ppm]", ylim=c(400, 425),
    cex.main = 1.5, cex.lab = 1.1, cex.axis =1.1)
points(c (1,2,6,12,20), TestData$co2[c (1,2,6,12,20)] , col="blue")
lines(c(1:20), estimated_y_hat_it)
lines(c(1:20), intervalll[1,], type = "l", lty=2, col = "red",  lwd = 2)
lines(c(1:20), intervalll[2,], type = "l", lty=2, col = "red",  lwd = 2)
legend("topleft", legend=c("predicted values","training data"),col=c( "red", "blue"),
      pch = c(1,1), cex=0.7)


############################

FinvN <- solve (F_new)
epsilon <- matrix(0, nrow = 1, ncol = 20)
sigma.scale.test<- matrix(0, nrow = 1, ncol = 20)
for ( k in 1:20 )
{

  sigma.scale.test[k] <- 1 + t(f(k)) %*% FinvN %*% f(k)
  epsilon[k]<- TestData$co2[k]-  t(f(k))%*%theta_new

}

plot(TrainData$time, TrainData$co2, col = "red",  ylab ="CO2", xlab = "time"
    , main = " test and training data together", ylim=c(250, 450))
points(TestData$time, TestData$co2)
first_theta<- theta_update[1,11:718]
firrst_theta <- rep(theta_new[1], times = 20)
lines(TrainData$time[11:718], first_theta,  col="green", lwd=2)
lines(TestData$time, firrst_theta, col="green",, lwd=2)

## Evaluate the best performing alpha=1-lambda
# (minimizing squared one step prediction errors) with a burnin period of 100 months.
# select to evaluate between these values [0:1] with 50 data points of precision.
lambda_test_4 <-seq(0.7,1,length=50)
estimated_y_hat_4 <- matrix(0, nrow= 50, ncol = 718)
pred_scale_4<- matrix(0, nrow = 1 , ncol=718)
theta_update_4<-matrix(0, nrow= 4, ncol = 718)
SSER <-matrix(0, nrow= 50, ncol = 718)
Y <-TrainData$co2
SSSS<-matrix(0, nrow= 1, ncol = 50)
Tt<- matrix(0,nrow = 1, ncol=1)
```

```r
interval_up<-matrix(0,nrow = 50, ncol=718)
interval_low<-matrix(0,nrow = 50, ncol=718)
for (k in 1:length(lambda_test_4))
{
  F_4 <- matrix(0, nrow=4, ncol=4)
  h_4 <- matrix(0,nrow=4,ncol=1)

  NN <- 10
  for (j in 0:(NN-1)){  #  first 10 observations as transient
    F_4 <- F_4 + lambda_test_4[k]^(j)*f(-j)%*%t(f(-j))
    h_4 <- h_4 + lambda_test_4[k]^(j)*f(-j)*Y[NN-j]
  }
  theta_update_4[,10]<- solve(F_4)%*% h_4

  SigmaInv_4 <-diag(lambda_test_4[k]^(N-(1:N)))

  #S_theta_N <- matrix(0, nrow= 1, ncol = 718)
  #Sig_local<-matrix(0, nrow= 1, ncol = 718)
  #interval <- matrix (0, nrow= 2, ncol=718)
  #SSig<-matrix(0, nrow= 1, ncol = 718)
  sigma.scale_4<- rep(NA, N)
  for (j in (NN+1):(N)) # start from N+1 = 11
  {
    sigma.scale_4[j] <- 1 + t(f(1)) %*% solve(F_4) %*% f(1)
    F_4<- F_4 + lambda_test_4[k]^(j-1)*f(-(j-1))%*%t(f(-(j-1)))
    h_4 <- lambda_test_4[k]*(LInv%*%h_4)+ f(0)%*%Y[j]
    theta_update_4[,j] <- solve(F_4)%*% h_4
    pred_scale_4[j]<- 1 + t(f(1))%*%solve(F_4)%*%f(1)

    estimated_y_hat_4[k,j]<- t(f(1))%*% theta_update_4[,j-1]

    Tt <- (1-lambda_test_4[k]^(j))/(1-lambda_test_4[k])
    interval_low[k,j] <- estimated_y_hat_4[k,j] + c(-1) * qt(0.975,Tt-4)* sqrt(pred_scale_4[j])
    interval_up[k,j] <- estimated_y_hat_4[k,j] + c(1) * qt(0.975,Tt-4)* sqrt(pred_scale_4[j])
    SSER[k,j] <- (Y[j]- estimated_y_hat_4[k,j])^2
  }
}
for (ite in 1:50)
{
  SSSS[ite] <-  sum(SSER[ite,101:718])
}

plot(lambda_test_4[17:47], (SSSS[17:47]), xlab=expression(lambda) , ylab="SSE", main =
expression(paste(" Evaluating the optimal ", lambda )) )
```

```
min(SSSS)
lambda_test_4[which(SSSS==min(SSSS))]

################################## Use the optimal lambda to predict
############################
# 1. Make a plot with the optimal l showing the data and predictions
# from 2010 and onwards (As in the previous question).
optimal_y_hat<-estimated_y_hat_4[which(SSSS==min(SSSS)),623:718]
plot (TrainData$time[623:718], optimal_y_hat, type = "l", lty=2, lwd = 2,
     ylab= "co2 [PPM]", xlab="time (years)" , main ="Predctions with optimal lambda")
lines( TrainData$time[623:718], TrainData$co2[623:718],lty=1, col="red", lwd = 2)
lines(TrainData$time[623:718], estimated_y_hat[623:718], lty=2,col="green", lwd = 2)
legend("topleft",legend = c("Optimal lambda - prediction", "Data", "lambda=0.9 prediction","conf
interval")
      ,col=c( "black", "red", "green", "blue"),lty=c(2,1,2,2),lwd=2, cex = 0.75)
lines(TrainData$time[623:718],interval_low[which(SSSS==min(SSSS)), 623:718], type = "l", lty=2,
col="blue" , lwd = 2)
lines(TrainData$time[623:718],interval_up[which(SSSS==min(SSSS)),623:718], type = "l", lty=2, col
= "blue",  lwd = 2)
################################## Future pred
###################################################
# Make a table predicting the test data as in the previous question.
```