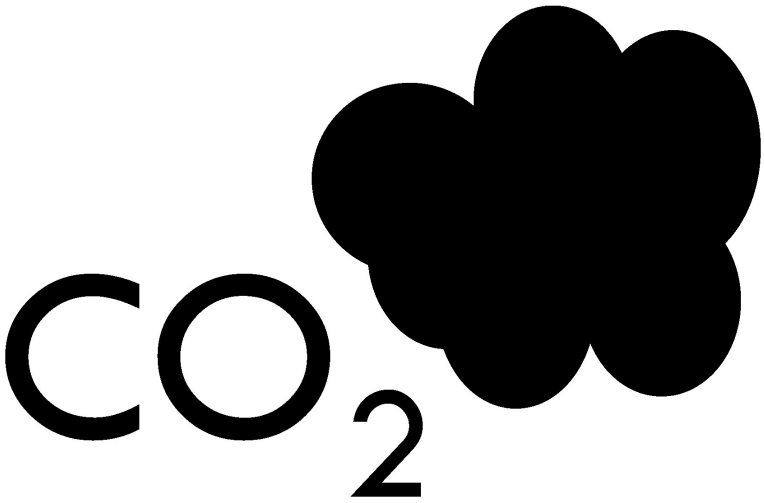
Danmarks Tekniske Universitet

DTU

Assignment 1: Atmospheric CO2

02417 - TIME SERIES ANALYSIS



GROUP

Pavlou, Ioannis - s212858
Blachet, Apolline - s222903
Kapakoglou, Georgios - s223001
Kozaris, Charalampos - s230224

February 23, 2023

Contents

List of Figures

List of Tables

1 Introduction

Time series analysis and forecasting are powerful tools for unlocking insights from complex environmental data and providing a basis for informed decision-making. By analyzing historical data on CO_2 concentrations in the atmosphere, it is possible to better understand how these levels are changing over time and how they may be influenced by factors such as human activity and natural processes.

The aim of this assignment was the estimation of different linear models, namely an "Ordinary Least Squares" (OLS) linear model, a "Weighted Least Squares" (WLS) linear model and a local linear trend model. For this purpose, a data set of monthly atmospheric CO_2 observations over the past 60 years from Mauna Loa, Hawaii was utilized. The data set was divided between training data, that was used for model training and parameter calibration, and testing data, used for testing of the models by comparing actual observations with model generated predictions.

For the construction of the linear models "RStudio" was employed. Firstly, the observations of CO_2 were plotted as a function of time. The OLS and WLS models were developed and tested with the testing data. The parameters of the models and the correlation coefficient ρ were estimated and the fitted values of both models were plotted together with the data for comparison.

Then, a local linear model with a given forgetting factor λ was introduced, corresponding to the linear model developed previously. The model was used to produce one-step predictions for the observations, errors and estimates of σ for each observation. Predictions were made for several time horizons and were compared to the testing data. One-step predictions for all observations and the estimated mean for each time step were plotted. Going one step further, an optimal forgetting factor λ was found by minimizing squared one step prediction errors. The predictions with the optimal λ were plotted and a table predicting the test data was created. Lastly, a qualitative performance comparison was made between the different models and extensions were suggested for future implementations.

2 Data description and visualization

An important step and crucial factor for a well structured statistical analysis is the preliminary data inspection and visualization, which is included in this section. The data set used consists of monthly measurements of atmospheric CO_2, taken at Manua Loa, Hawaii in a period of 60 years. The data is provided by NOAA/ESRL and can be accessed at: https://gml.noaa.gov/ccgg/trends/.

The data set is comprised by 738 observations and 4 attributes depicted in Table 1:

Table 1: Data attributes

year	the year measurement was made
month	the month measurement was made
time	decimal year and month
co2	atmospheric CO_2 concentration [ppm]

In order to train the different models it was essential to split the provided data set into training and test data sets. A safe split would be to utilize 70% of the existing data set for training and 30% for testing. But due to the relative small size of available observations approximately 97.29 $\%$ of the data have been used for training and only 2.71 $\%$ for testing. More specifically, the observations up to 2017 were used as the training set and the observations for years 2018 and 2019 were used as the testing set. Remarkably, there were no missing values or potential outliers. In order to visualize the above, in Figure 1 the observations of the CO_2 levels are plotted as a function of time and in Figure 2 the range of the observed values is illustrated as a boxplot.

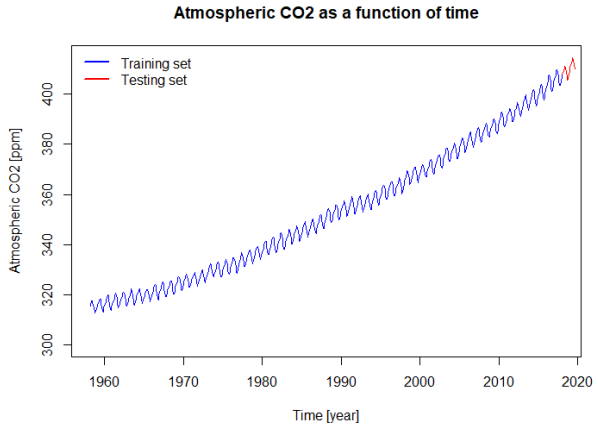


Figure 1: Visualization of raw data

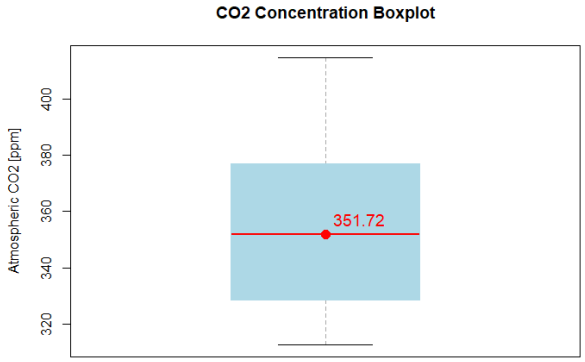


Figure 2: Boxplot of CO_2 concentration

It is evident from Figure 1 that CO_2 levels show a clear increasing trend, with an increase of $\sim 33\%$ for the reference period. Furthermore, the range of CO2 concentration values above the median is higher than the range of values below the median as observed in Figure 2. Conjointly, it is implied a higher rate of increase the last two decades.

3 OLS and WLS

In this section two different versions for estimating the parameters of the same general linear model are described. Their difference is that in the "Ordinary Least Squares" (OLS) method of estimating the parameters the same weight is given in all observations in contrast to the "Weighted Least Squares" (WLS) method on which different weights are put to different observations.

Firstly, the OLS model was developed and tested. The model was of the form:

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

where t denotes the time step for the Y_t, α is the intercept, β_t, β_t and β_c are the slope parameters of the model, p is the period of the annual harmonic part and ϵ_t is a sequence of random variables with $E[\epsilon_t] = 0$ and $\mathrm{Var}[\epsilon_t] = \sigma_t^2$.

The period of the harmonic part p was set to 1 (1 year), in order to represent the periodicity that data illustrate in Figure 3. At this point, it shall be mentioned that the unit of time was chosen to be 1 year for the whole assignment, which means that we moved 1/12 of year between observations. The design matrices for the training and testing sets were constructed, including the Fourier term to capture the pre-mentioned periodicity in the data and the OLS estimates of the regression coefficients were computed. Lastly, the predicted values of CO_2 concentration and the residuals for the training and testing sets were computed.

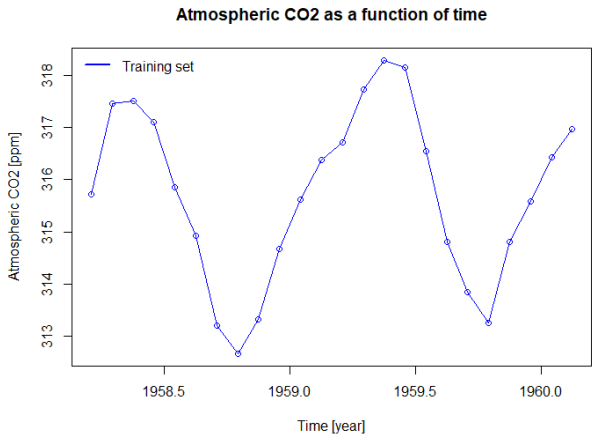


Figure 3: Harmonic period indicating ρ : 1 year

For WLS, $\mathrm{Var}[\epsilon_t] = \sigma_t^2\Sigma$, where Σ is the correlation matrix. The correlation structure of the residuals was considered to be an exponential decaying function of the time distance between two observations and the optimal correlation coefficient was considered estimated using 5 iterations of the relaxation algorithm and was found to be equal to ρ = *0.9822406*. The correlation matrix was calculated as follows:

$$\Sigma = \begin{bmatrix} 1.0000000 & 0.9822406 & 0.9647966 & \dots \\ 0.9822406 & 1.0000000 & 0.9822406 & \dots \\ 0.9647966 & 0.9822406 & 1.0000000 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2}$$

The measure of uncertainty for each of the estimates and the MSE calculated on the training set and the testing set are presented in Table 2. Lastly, the OLS fit and the WLS fit were added to the plot, as it is shown in Figure 4 and Figure 5 respectively.

Table 2: Estimated parameters in OLS and WLS models with their standard deviations

parameter	**OLS**	**WLS**
α	-2709.670 ($\pm$14.996)	-2743.980 ($\pm$128.394)
β_t	+1.540 ($\pm$0.008)	+1.558 ($\pm$0.065)
β_s	+2.614 ($\pm$0.184)	+2.648 ($\pm$0.068)
β_c	-1.050 ($\pm$0.184)	-1.019 ($\pm$0.068)
MSE (train set)	12.117	13.354
MSE (test set)	86.320	59.045

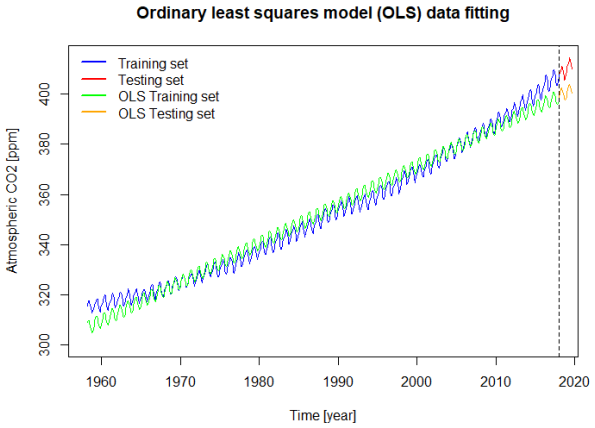


Figure 4: Data fitting of OLS linear regression model

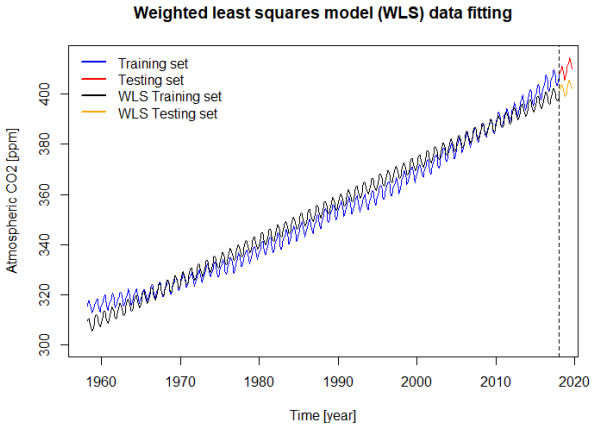


Figure 5: Data fitting of WLS linear regression model

To sum up, from Table 2 it can be observed that the OLS and WLS models have similar performances on the training set. However, the Mean Squared Error (MSE) of the WLS model is significantly lower in the test data which consequently implies a better model performance as the predicted values are closer to the observations. Thus, the WLS model can perform better on new data.

4 Local linear trend model

In this section a local linear trend model with a forgetting factor $\lambda = 0.9$ was developed to estimate one step predictions. The matrix L and the $f(0)$ for the trend model corresponding to the linear model in the previous section, respectively are:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.0833 & 1 & 0 & 0 \\ 0 & 0 & 0.866 & 0.5 \\ 0 & 0 & -0.5 & 0.866 \end{bmatrix} \quad \text{and} \quad f(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{3}$$

The data were filtered with the chosen model and the one step predictions were estimated based on the following equations.

$$\begin{aligned} F_{N+1} &= F_N + \lambda^N \cdot f(-N) \cdot f^T(-N) \\ h_{N+1} &= \lambda L^{-1} h_N + f(0) Y_{N+1} \\ \hat{\theta}_{N+1} &= F_{N+1}^{-1} h_{N+1} \end{aligned} \tag{4}$$

In order to assess the performance of the model, the one step prediction errors and their respective σ were calculated. The results are illustrated in Figure 6 (10 first observations were skipped as transient). We observe for the training set that the residuals are i.i.d. (independent identically distributed) and the resulting estimates of σ for each observation are pretty similar.

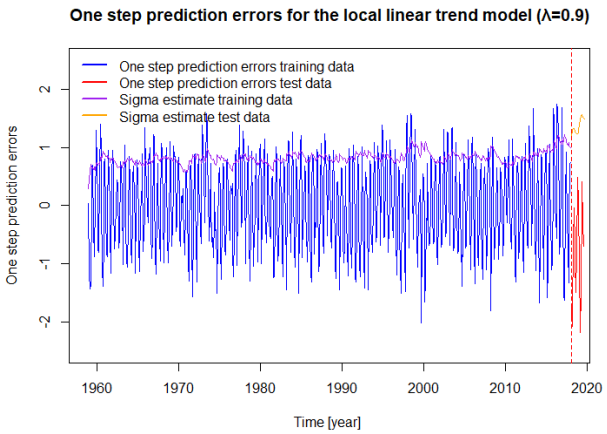


Figure 6: One step prediction errors and the resulting estimates of σ

In Figure 6 it appears that the standard deviation is increasing for the one step predictions of the test data set. This implies the expected increase of uncertainty for new data and future predictions. More specifically, the predicted values tend to be overestimated and that is why there is a negative increase in residual values. However, the variance of the residuals is similarly distributed for the future predictions, but with the mean shifted from approximately 0 to -1. That could also

be taken into consideration for future model adjustments in order to account for systematic errors.

In Figures 7 and 8 are illustrated the one-step predictions of the local trend linear model, where the blue line represents the actual time series data, the green line represents the model's one-step predictions, and the shaded area represents a 95% prediction interval. The first plot shows the one-step prediction errors for the entire data set, which has been separated into a training set and a test set. The second plot zooms in on a specific period of time to show the details of the one-step prediction errors. The one-step predictions and their 95% prediction intervals are generally close to the actual observations, indicating that the model is doing a good job of accurately predicting the future values of the time series both for the training and the test set.

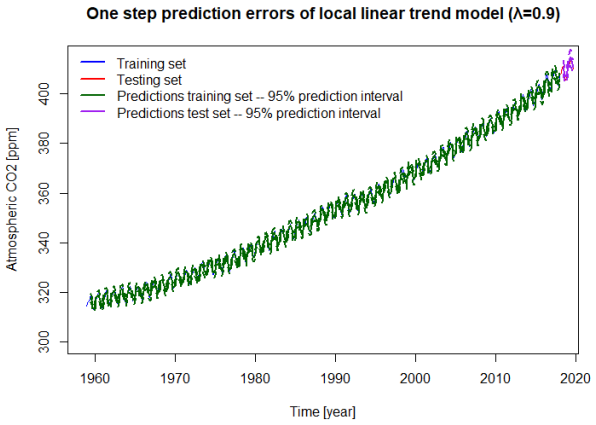


Figure 7: One step predictions - whole period

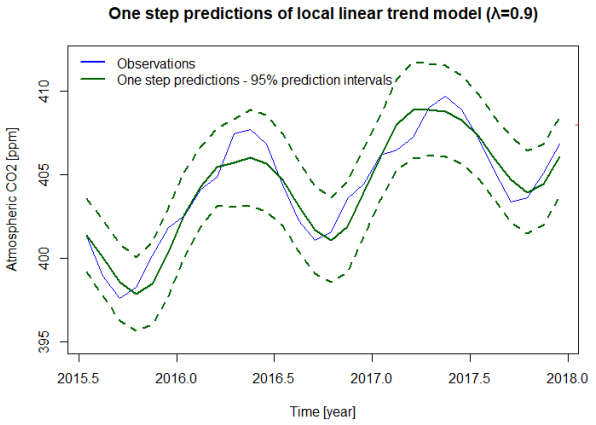


Figure 8: Zoom in (2015.5-2018)

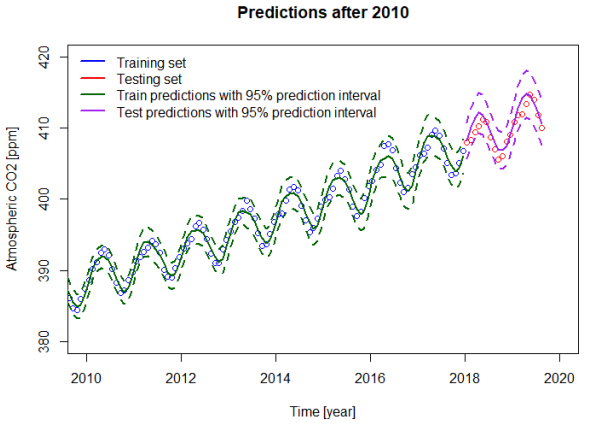


Figure 9: Model one step predictions since 2010

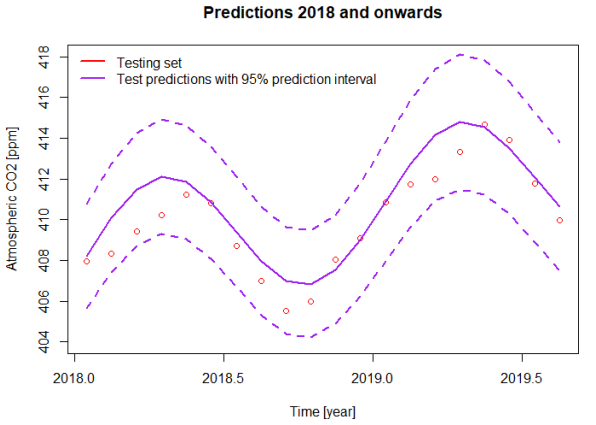


Figure 10: Zoom in (test data)

In Figure 9 are illustrated the time series data since 2010, along with one-step ahead predictions for the train and test data using a local trend linear model. The blue line represents the actual time series training data, while the green line represents the model's one-step ahead predictions for

the training data. The shaded region around the green line represents the 95% prediction interval, which widens as we move further into the future, indicating greater uncertainty in the model's predictions. Zooming in, in Figure 10 the one-step predictions for the test data are generally close enough to the actual observations, indicating that the model is accurately capturing the underlying patterns of the data. Overall, the local trend linear model appears to be a useful tool for predicting the future values of the time series, especially in the short-term. However, it shall be stated that further analysis may be needed to improve its accuracy over longer time horizons.

In Table 3 are illustrated the predictions for 1,2,6,12 and 20 months ahead, as well as their difference with the observed data. Overall, the MSE for the test data is $MSE = 1.232$, which indicates a really good local model fit.

Table 3: Predictions 1,2,6,12 and 20 months ahead

Months ahead	1	2	6	12	20
Observations	407.96	408.32	410.79	409.07	409.95
Predictions	408.21	410.08	410.83	409.00	410.65
Obsv. - Pred.	-0.25	-1.76	-0.04	0.07	-0.7

In Figure 11 the data and the estimated mean by the local linear trend model for each time step are plotted (which is typically the first element in θ_t for all t). Based on that, it can be observed that the local linear trend model is able to capture the overall trend in the data as it fluctuates over time. The first element of the parameter vector θ_t for the local linear trend model is the estimated level at time t, while the rest of the elements are the estimated slopes at time t. The estimated mean for each time step which is equivalent to the estimated level at that time step, closely follows the general shape of the data indicating that the model is a good fit. Apparently, for the test set it remains constant as it is not updated. Overall, the model is able to track the upward trend in the data with only a few small fluctuations.

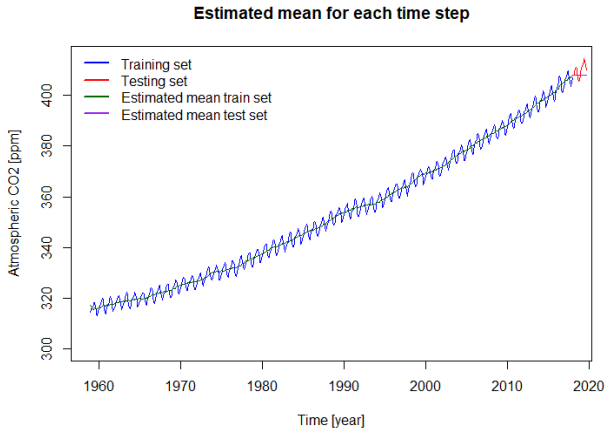


Figure 11: Estimated mean for each time step

5 Optimal λ

In this section it was explored the optimal λ value by minimizing the squared one step prediction errors with a burning period of 100 years. It shall be mentioned that the burning period in an optimization process is a preliminary period during which the system is allowed to settle into its steady-state behavior. The purpose of the burning period is to ensure that the optimization algorithm starts with a representative set of data points, and to avoid using the transient data that may distort the optimization results. The length of the burning period is typically determined by the time it takes for the system to reach its steady-state behavior, and it may vary depending on the specific system being modeled and the optimization method being used.

In Figure 12 it is depicted the optimization process of λ value, which has converged to the value of: $\lambda = 0.969$. Furthermore, in Figure 13 are illustrated the data and the local linear trend model predictions from 2010 and onwards with the optimal λ. Lastly, in Table 4 are illustrated the predictions for 1,2,6,12 and 20 months ahead with the optimal λ, as well as their difference with the observed data. Overall, the MSE for the test data is $MSE = 0.677$, which indicates a better local model fit than the base case scenario where λ=0.9. It shall be mentioned that as burning period was used a period of 100 months exactly before 2010 in order to capture the faster upward trend behavior of the last two decades.

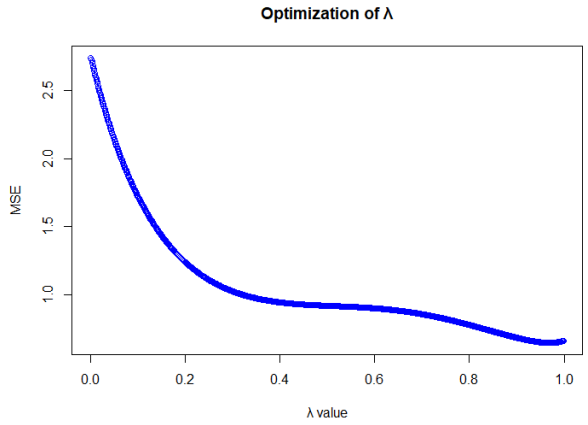


Figure 12: Optimization process of λ

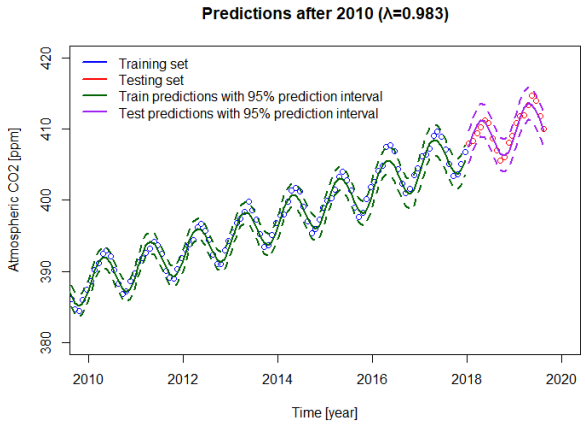


Figure 13: Predictions since 2010 (λ=0.983)

Table 4: Predictions 1,2,6,12 and 20 months ahead (λ=0.983)

Months ahead	1	2	6	12	20
Observations	407.96	408.32	410.79	409.07	409.95
Predictions	407.49	409.24	410.14	408.11	409.79
Obsv. - Pred.	0.47	-0.92	0.65	0.96	0.16

6 Conclusions and further extensions

The fact that we evaluate the various models' performance with only 20 observations is indeed a limitation of the performed statistical analysis. Different cross validation methods shall be applied, like a 3-fold cross validation method or the simpler holdout method, in which 70% of the existing observations comprise the training set and 30% the test set. Additionally, the different models' performance shall be evaluated for various time resolution and aggregation levels of the training data. This way, the robustness of each model's performance will be enhanced, overfitting can be avoided and better confidence and prediction intervals can be estimated.

Regarding the optimization of the forgetting factor λ, it was observed that the estimated optimal value depends upon the period that is selected to be used as a burning period. If the first 100 months of observations are used, the optimal estimated value is $\lambda = 0.983$, with a MSE = 2. If on the other hand the 100 observations preceding 2010 are chosen to serve as the burning period, the optimal estimated value is $\lambda = 0.969$, with a MSE = 0.677, significantly lower than the previous value. This is the result of the linear model's local nature, meaning that when the burning period is closer to the period we are interested in making a forecast about, the model is able to capture local trends more efficiently, resulting in better predictions. As demonstrated in Figure 12, a local minimum can be observed in the MSE curve, leading to the conclusion that the optimal value for λ has indeed been estimated, at least at a local level.

Among the models developed in this assignment, the best choice depends on the prediction time horizon of interest. If the goal of the forecast is to predict accurate values for a relatively short future period, the local linear trend model with an optimal forgetting factor would be the best choice, as it produces the most accurate estimations of the future values. As the time period that we consider increases, different trends may arise in the data, affecting the explained variance of the residuals of the predictions. That would lead to over- or underestimation of the actual values, thus radically downgrading the model's performance. In this case, a WLS model would be a safer choice for predictions in the long term, because at the expense of a trade-off between flexibility and accuracy, in the end an overall better estimation of the data's projected trajectory is to be expected.

Possible extensions of the model could include the use of a different loss function for the estimation of the optimal λ for the local linear trend model, and/or the use of different optimization methods. All in all, there is additional space for further exploration and analysis. Apart from the "General Linear Model" (GLM), a machine learning approach to predict the target variable (response variable) based on the the features of the model could be utilized, such as a "Decision Tree algorithm", but this is out of the scope of this project.

Appendix

Code

Listing 1: Preliminary data analysis and visualization

```r
# Load the data
data <- read.table("A1_co2.txt", header = TRUE)

# Split the data into training and testing sets
xtrain <- data$time[1:718]
ytrain <- data$co2[1:718]
xtest <- data$time[719:738]
ytest <- data$co2[719:738]

# Get the number of observations in training set
ntrain <- length(xtrain)

# Plot the training and testing sets
par(mfrow=c(1,1))
plot(xtrain, ytrain, type="l", col="blue",
     xlab="Time [year]", ylab="Atmospheric CO2 [ppm]",
     main="Atmospheric CO2 as a function of time", ylim=c(300, max
         (data$co2)))
lines(xtest, ytest, col="red")
legend("topleft", c("Training set", "Testing set"), col=c("blue",
   "red"), lty=1, bty='n', lwd=2)

# Boxplot of CO2 concentration
par(mfrow=c(1,1))
boxplot(data$co2, main="CO2 Concentration Boxplot", ylab="
   Atmospheric CO2 [ppm]", col="lightblue",
        medcol="red", medlwd=2, medpch=19, medcex=1.5,
        boxcol="lightblue", whiskcol="darkgray")
# Add median value to the plot
text(1, median(data$co2) + 5, round(median(data$co2),2), col="red"
    , pos=4, cex=1.2)
```

Listing 2: Design matrix

```r
# Check one harmonic period
par(mfrow=c(1,1))
plot(xtrain[1:24], ytrain[1:24], type="o", col="blue",
     xlab="Time [year]", ylab="Atmospheric CO2 [ppm]",
```

```
5       main="Atmospheric CO2 as a function of time", ylim=c(min(data
            $co2[1:24]), max(data$co2[1:24])))
6   lines(xtest, ytest, col="red")
7   legend("topleft", c("Training set"), col=c("blue"), lty=1, bty='n'
        , lwd=2)
8
9   # Set the period of the harmonic part
10  p <- 1 # we've set p to 1 because we can see from the observations
        that the period of the harmonic part is approximately 1 (1
        year)
11
12  # Construct the design matrices for the training and testing sets,
        including Fourier terms to capture periodicity in the data
13  Xtrain <- cbind(1, xtrain, sin(2*pi*xtrain/p), cos(2*pi*xtrain/p))
14  Xtest <- cbind(1, xtest, sin(2*pi*xtest/p), cos(2*pi*xtest/p))
```

Listing 3: OLS

```
1   # Compute the OLS estimate of the regression coefficients
2   thetahatOLS <- solve(t(Xtrain) %*% Xtrain) %*% t(Xtrain) %*%
        ytrain
3
4   # Compute the predicted values of CO2 concentration for the
        training and testing sets
5   yhatOLS_train <- Xtrain %*% thetahatOLS
6   yhatOLS_test <- Xtest %*% thetahatOLS
7
8   # Compute residual for the training and testing sets
9   epsOLS_train <- ytrain - yhatOLS_train
10  epsOLS_test <- ytest - yhatOLS_test
11
12  # Compute measure of uncertainty for each of the estimates
13  sigma2OLS <- t(epsOLS_train) %*% epsOLS_train / (length(xtrain) -
        length(thetahatOLS))
14  varOLS <- sigma2OLS[1] * solve(t(Xtrain) %*% Xtrain)
15  stdOLS <- sqrt(diag(varOLS))
16
17  # Compute MSE
18  MSE_OLS_train <- t(epsOLS_train) %*% epsOLS_train / length(epsOLS_
        train)
19  MSE_OLS_test <- t(epsOLS_test) %*% epsOLS_test / length(epsOLS_
        test)
20
21  # Plot the training and testing sets
```

```
22 par(mfrow=c(1,1))
23 plot(xtrain, ytrain, type="l", col="blue",
24      xlab="Time [year]", ylab="Atmospheric CO2 [ppm]",
25      main="Ordinary least squares model (OLS) data fitting", ylim=
           c(300, max(data$co2)))
26 lines(xtest, ytest, col="red")
27
28 # Add the OLS fit to the plot
29 lines(xtrain, yhatOLS_train, col="green")
30 lines(xtest, yhatOLS_test, col="orange")
31 legend("topleft", c("Training set", "Testing set", "OLS Training
      set", "OLS Testing set"), col=c("blue", "red", "green", "orange
      "), lty=1, bty='n', lwd=2)
32 abline(v=xtest[1], col="black", lty=2)
```

Listing 4: WLS

```
1  # Initial guess of correlation structure
2  rau <- 0.8
3  Sigmay <- diag(ntrain)
4  for (i in 1:ntrain) {
5      for (j in 1:ntrain) {
6          Sigmay[i,j] <- rau^abs(i-j)
7      }
8  }
9
10 for(i in 1:5){
11      # Estimate parameters using currently assumed correlation
           structure
12      thetahatWLS <- solve(t(Xtrain) %*% solve(Sigmay) %*% Xtrain)
           %*% (t(Xtrain) %*% solve(Sigmay) %*% ytrain)
13
14      # Compute residuals for these parameter estimates
15      epsWLS <- ytrain - Xtrain %*% thetahatWLS
16
17      # Select the value for Sigamy which reflects the correlation
           and variance structure
18      rau <- cor(epsWLS[1:ntrain-1], epsWLS[2:ntrain])
19      for (i in 1:ntrain) {
20          for (j in 1:ntrain) {
21              Sigmay[i,j] <- rau^abs(i-j)
22          }
23      }
24 }
```

```
25
26 # Compute the predicted values of CO2 concentration for the
       training and testing sets
27 yhatWLS_train <- Xtrain %*% thetahatWLS
28 yhatWLS_test <- Xtest %*% thetahatWLS
29
30 # Compute residual for the training and testing sets
31 epsWLS_train <- ytrain - yhatWLS_train
32 epsWLS_test <- ytest - yhatWLS_test
33
34 # Compute measure of uncertainty for each of the estimates
35 sigma2WLS <- t(epsWLS_train) %*% solve(Sigmay) %*% epsWLS_train /
       (length(xtrain) - length(thetahatWLS))
36 varWLS <- sigma2WLS[1] * solve(t(Xtrain) %*% solve(Sigmay) %*%
       Xtrain)
37 stdWLS <- sqrt(diag(varWLS))
38
39 # Compute MSE
40 MSE_WLS_train <- t(epsWLS_train) %*% epsWLS_train / length(epsWLS_
       train)
41 MSE_WLS_test <- t(epsWLS_test) %*% epsWLS_test / length(epsWLS_
       test)
42
43 # Plot the training and testing sets
44 par(mfrow=c(1,1))
45 plot(xtrain, ytrain, type="l", col="blue",
46      xlab="Time [year]", ylab="Atmospheric CO2 [ppm]",
47      main="Weighted least squares model (WLS) data fitting", ylim=
           c(300, max(data$co2)))
48 lines(xtest, ytest, col="red")
49
50 # Add the WLS fit to the plot
51 lines(xtrain, yhatWLS_train, col="Black")
52 lines(xtest, yhatWLS_test, col="Orange")
53 legend("topleft", c("Training set", "Testing set", "WLS Training
       set", "WLS Testing set"), col=c("blue", "red", "black", "orange
       "), lty=1, bty='n', lwd=2)
54 abline(v=xtest[1], col="black", lty=2)
```

Listing 5: Q.1.3.1-2: Data filtering with the local linear trend model

```
1 ## Local linear trend model for the data.
2 lambda <- 0.9
3 f <- function(j) rbind(1, j, sin(2*pi*j/p), cos(2*pi*j/p))
```

```
L <- t(matrix(c(1,0,0,0, 1/12,1,0,0, 0,0,cos(2*pi/p/12),sin(2*pi/p
    /12), 0,0,-sin(2*pi/p/12),cos(2*pi/p/12)),ncol=4))
LInv <- solve(L)

# 4 parameters so at 4 observations are needed to estimate theta
# however one more is needed to get an estimate of the uncertainty

init <- 5 # Skip estimating for the first 10 observations
## FNinit & hNinit (First observations) using equation (3.100)
F <- matrix(0, nrow=4, ncol=4)
h <- matrix(0,nrow=4,ncol=1)
for (j in 0:(init-1)){
  F <- F + lambda^(j) * f(-j/12) %*% t(f(-j/12))
  h <- h + lambda^(j) * f(-j/12) * ytrain[init-j]
}

## Allocating space
np <- length(h)
theta.all <- matrix(NA,ncol=np, nrow=ntrain)
sigma.all <- rep(NA, ntrain)
sd.err.all <- rep(NA, ntrain)
yhat.all <- rep(NA, ntrain)
err.all <- rep(NA, ntrain)

## Solving at time init
theta.hat <- solve(F, h)
theta.all[init,] <- theta.hat
epsilon <- ytrain[1:init] - cbind(1, (-(init-1):0)/12, sin(2*pi*
    (-(init-1):0)/12/p), cos(2*pi*(-(init-1):0)/12/p)) %*% theta.
    hat

T <- 0
for (j in 0:(init-1)){
     T <- T + lambda^(j)
}

Sigma_inv <- diag(init)
for (j in 0:(init-1)){
     Sigma_inv[j,j] <- lambda^((init-1-j))
}

sigma.all[init] <- sqrt(t(epsilon) %*% Sigma_inv %*% epsilon/(T -
    np))
sd.err.all[init] <- sigma.all[init] * sqrt(1 + t(f(1/12)) %*%
```

```
    solve(F) %*% f(1/12))
yhat.all[init+1] <- t(f(1/12)) %*% theta.hat
err.all[init+1] <- ytrain[init+1] - yhat.all[init+1]

## Looping over the remaining observations
for (i in (init+1):(ntrain)){
  F <- F + lambda^((i-1)) * f(- (i-1)/12) %*% t(f(- (i-1)/12))
  h <- lambda * LInv %*% h + f(0)*ytrain[i]
  theta.hat <-  solve(F, h)
  theta.all[i,] <- theta.hat

  ## Adding uncertainty information
  epsilon <- ytrain[1:i] - cbind(1, (-(i-1):0)/12, sin(2*pi*(-(i
     -1):0)/12/p), cos(2*pi*(-(i-1):0)/12/p)) %*% theta.hat
  T <- 0
  for (j in 0:(i-1)){
     T <- T + lambda^(j)
  }

  Sigma_inv <- diag(i)
  for (j in 0:(i-1)){
     Sigma_inv[j,j] <- lambda^((i-1-j))
  }

  sigma.all[i] <- sqrt(t(epsilon) %*% Sigma_inv %*% epsilon/(T -
     np))

  ## Estimating s.d. of estimated parameters
  sd.err.all[i] <- sigma.all[i] * sqrt(1 + t(f(1/12)) %*% solve(F)
      %*% f(1/12))

  yhat.all[i+1] <- t(f(1/12)) %*% theta.hat
  err.all[i+1] <- ytrain[i+1] - yhat.all[i+1]
}

## Predictions on test set
theta_pred <- theta.all[718,] # Get last theta of the training set
sigma_pred <- sigma.all[718] # Get last sigma

ntest <- length(xtest)

# Allocate memory
yhat_test.all <- rep(NA, ntest)
err.test.all <- rep(NA, ntest)
```

```
sd.err_test.all <- rep(NA, ntest)

for (j in 1:ntest){
      yhat_test.all[j] <- t(f(j/12)) %*% theta_pred # Make
         prediction
      sd.err_test.all[j] <- sigma_pred * sqrt(1 + t(f(j/12)) %*%
         solve(F) %*% f(j/12)) # Calculate standard deviation of
         the error
      err.test.all[j] <- ytest[j+1] - yhat_test.all[j+1]
}
```

Listing 6: Q.1.3.3-8: Various plots

```
## Plot Q3.3 ##
par(mfrow=c(1,1))
plot(xtrain[10:length(xtrain)], err.all[10:length(xtrain)], type="
   l", col="blue",
      xlab="Time [year] ", ylab="One step prediction errors",
      main="One step prediction errors for the local linear trend
         model (Îť=0.9)",
      ylim=c(-2.5, 2.5))
lines(xtest, err.test.all, type="l", col="red")
lines(xtrain[10:length(xtrain)], sigma.all[10:length(xtrain)], col
   ='purple')
lines(xtest, sd.err_test.all, col='orange')
legend("topleft", c("One step prediction errors trainind data", "
   One step prediction errors test data","Sigma estimate training
   data","Sigma estimate test data"), col=c("blue", "red","purple"
   , "orange"), lty=1, bty='n', lwd=2)
abline(v=xtest[1], col="red", lty=2)

## Plot Q3.4 ##
t.quan <- qt(p = 0.975, df=rep(ntrain-np, ntrain-9)) # Get t
   distribution for training
t.quan[1:init] <- NA

t_test.quan <- qt(p = 0.975, df=rep(ntest-np, ntest))
t_test.quan[1:init] <- NA

par(mfrow=c(1,1))
plot(xtrain[10:length(xtrain)], ytrain[10:length(xtrain)], type="l
   ", col="blue",
      xlab="Time [year]", ylab="Atmospheric CO2 [ppm]",
      main="One step prediction errors of local linear trend model
```

```
        (Ît=0.9)", ylim=c(300, max(data$co2)))
lines(xtest, ytest, col="red")
#lines(xtrain[10:length(xtrain)], yhat.all[10:length(xtrain)],
    col="darkgreen")
matlines(xtrain[10:length(xtrain)], yhat.all[10:length(xtrain)] +
    t.quan * cbind(0,-sd.err.all[10:length(xtrain)],sd.err.all[10:
    length(xtrain)]),type="l",lty=c(1,2,2),lwd=2, col="darkgreen")
matlines(xtest, yhat_test.all + t_test.quan * cbind(0,-sd.err_test
    .all,sd.err_test.all),type="l",lty=c(1,2,2),lwd=2, col="purple"
    )
legend("topleft", c("Training set", "Testing set", "Predictions
    training set — 95% prediction interval", "Predictions test set
     — 95% prediction interval"), col=c("blue", "red", "darkgreen"
    , "purple"), lty=1, bty='n', lwd=2)

## Plot Q3.5 ##
t.quan_test <- qt(p = 0.975, df= rep(ntest-np, ntest)) # Get t
    distribution for testing

par(mfrow=c(1,1))
plot(xtrain[10:length(xtrain)], ytrain[10:length(xtrain)], col="
    blue",
      xlab="Time [year]", ylab="Atmospheric CO2 [ppm]",
      main="Predictions after 2010",
      xlim=c(2010, 2020), ylim=c(380, 420))
points(xtest, ytest, col="red")
matlines(xtrain[10:length(xtrain)], yhat.all[10:length(xtrain)] +
    t.quan * cbind(0,-sd.err.all[10:length(xtrain)],sd.err.all[10:
    length(xtrain)]),type="l",lty=c(1,2,2),lwd=2, col="darkgreen")
matlines(xtest, yhat_test.all + t.quan_test * cbind(0,-sd.err_test
    .all, sd.err_test.all),type="l",lty=c(1,2,2),lwd=2, col="purple
    ")
legend("topleft", c("Training set", "Testing set", "Train
    predictions with 95% prediction interval", "Test predictions
    with 95% prediction interval"), col=c("blue", "red", "darkgreen
    ", "purple"), lty=1, bty='n', lwd=2)

# Zoom in test data
par(mfrow=c(1,1))
plot(xtest, ytest, col="red",
      xlab="Time [year]", ylab="Atmospheric CO2 [ppm]",
      main="Predictions 2018 and onwards",
      ylim=c(404, 418))
matlines(xtest, yhat_test.all + t.quan_test * cbind(0,-sd.err_test
```

```
    .all, sd.err_test.all),type="l",lty=c(1,2,2),lwd=2, col="purple
    ")
50 legend("topleft", c("Testing set", "Test predictions with 95%
    prediction interval"), col=c("red", "purple"), lty=1, bty='n',
    lwd=2)
51
52 ## Question 1.3.6: Predictions 1, 2, 6, 12 and 20 months ahead &
    Question 1.3.7
53 j.all <- c(1,2,6,12,20)
54 yhat_pred.all <- rep(NA, length(j.all))
55 dif.pred <- rep(NA, length(j.all))
56
57 for (j in 1:length(j.all)){
58      yhat_pred.all[j] <- t(f(j.all[j]/12)) %*% theta_pred # Make
        prediction
59 }
60
61 MSE <- sum((ytest-yhat_test.all)^2)/ntest
62
63 ## Plot Q3.8 ##
64 par(mfrow=c(1,1))
65 plot(xtrain[10:length(xtrain)], ytrain[10:length(xtrain)], type="l
    ", col="blue",
66      xlab="Time [year]", ylab="Atmospheric CO2 [ppm]",
67      main="Estimated mean for each time step", ylim=c(300, max(
        data$co2)))
68 lines(xtest, ytest, col="red")
69 lines(xtrain[10:length(xtrain)], theta.all[10:length(xtrain),1],
    col='darkgreen')
70 lines(xtest, rep(theta.hat[1], length(xtest)), col='purple')
71 legend("topleft", c("Training set", "Testing set", "Estimated mean
     train set", "Estimated mean test set"), col=c("blue", "red", "
    darkgreen", "purple"), lty=1, bty='n', lwd=2)
```

Listing 7: Q.1.4: Optimal λ

```
1 ### Find optimal lambda ###
2
3 # Extract burning period of 100 months
4 xtrain <- data$time[523:622]
5 ytrain <- data$co2[523:622]
6
7 xtest <- data$time[719:738]
8 ytest <- data$co2[719:738]
```

```
9
10 # Get the number of observations in training set
11 ntrain <- length(xtrain)
12
13 ## Construct design matrix ##
14
15 # Set the period of the harmonic part
16 p <- 1 # we've set p to 1 because we can see from the observations
       that the period of the harmonic part is approxiately 1 (1 year
   )
17
18 # Construct the design matrices for the training and testing sets,
       including Fourier terms to capture periodicity in the data
19 Xtrain <- cbind(1, xtrain, sin(2*pi*xtrain/p), cos(2*pi*xtrain/p))
20 Xtest <- cbind(1, xtest, sin(2*pi*xtest/p), cos(2*pi*xtest/p))
21
22
23 ## Local linear trend model for the data.
24 f <- function(j) rbind(1, j, sin(2*pi*j/p), cos(2*pi*j/p))
25 L <- t(matrix(c(1,0,0,0, 1/12,1,0,0, 0,0,cos(2*pi/p/12),sin(2*pi/p
   /12), 0,0,-sin(2*pi/p/12),cos(2*pi/p/12)),ncol=4))
26 LInv <- solve(L)
27
28 lambdas <- seq(from = 0.001, to = 0.9999, by = 0.001)
29 MSE.all <- rep(NA, length(lambdas))
30 c <- 1
31 for (lambda in lambdas){
32
33 # 4 parameters so at 4 observations are needed to estimate theta
34 # however one more is needed to get an estimate of the uncertainty
35
36 init <- 5 # Skip estimating for the first 10 observations
37 ## FNinit & hNinit (First observations) using equation (3.100)
38 F <- matrix(0, nrow=4, ncol=4)
39 h <- matrix(0,nrow=4,ncol=1)
40 for (j in 0:(init-1)){
41   F <- F + lambda^(j) * f(-j/12) %*% t(f(-j/12))
42   h <- h + lambda^(j) * f(-j/12) * ytrain[init-j]
43 }
44
45 ## Allocating space
46 np <- length(h)
47 theta.all <- matrix(NA,ncol=np, nrow=ntrain)
48 sigma.all <- rep(NA, ntrain)
```

```
49 sd.err.all <- rep(NA, ntrain)
50 yhat.all <- rep(NA, ntrain)
51 err.all <- rep(NA, ntrain)
52
53 ## Solving at time init
54 theta.hat <- solve(F, h)
55 theta.all[init,] <- theta.hat
56 epsilon <- ytrain[1:init] - cbind(1, (-(init-1):0)/12, sin(2*pi*
       (-(init-1):0)/12/p), cos(2*pi*(-(init-1):0)/12/p)) %*% theta.
       hat
57
58 T <- 0
59 for (j in 0:(init-1)){
60      T <- T + lambda^(j)
61 }
62
63 Sigma_inv <- diag(init)
64 for (j in 0:(init-1)){
65      Sigma_inv[j,j] <- lambda^((init-1-j))
66 }
67
68 sigma.all[init] <- sqrt(t(epsilon) %*% Sigma_inv %*% epsilon/(T -
       np))
69 sd.err.all[init] <- sigma.all[init] * sqrt(1 + t(f(1/12)) %*%
       solve(F) %*% f(1/12))
70 yhat.all[init+1] <- t(f(1/12)) %*% theta.hat
71 err.all[init+1] <- ytrain[init+1] - yhat.all[init+1]
72
73 ## Looping over the remaining observations
74 for (i in (init+1):(ntrain)){
75   F <- F + lambda^((i-1)) * f(-(i-1)/12) %*% t(f(-(i-1)/12))
76   h <- lambda * LInv %*% h + f(0)*ytrain[i]
77   theta.hat <- solve(F, h)
78   theta.all[i,] <- theta.hat
79
80   ## Adding uncertainty information
81   epsilon <- ytrain[1:i] - cbind(1, (-(i-1):0)/12, sin(2*pi*(-(i
       -1):0)/12/p), cos(2*pi*(-(i-1):0)/12/p)) %*% theta.hat
82   T <- 0
83   for (j in 0:(i-1)){
84      T <- T + lambda^(j)
85   }
86
87   Sigma_inv <- diag(i)
```

```
88    for (j in 0:(i-1)){
89      Sigma_inv[j,j] <- lambda^((i-1-j))
90    }
91
92    sigma.all[i] <- sqrt(t(epsilon) %*% Sigma_inv %*% epsilon/(T -
          np))
93
94    ## Estimating s.d. of estimated parameters
95    sd.err.all[i] <- sigma.all[i] * sqrt(1 + t(f(1/12)) %*% solve(F)
          %*% f(1/12))
96
97    yhat.all[i+1] <- t(f(1/12)) %*% theta.hat
98    err.all[i+1] <- ytrain[i+1] - yhat.all[i+1]
99
100 }
101
102 # Compute and store MSE
103 MSE <- (t(err.all[(init+1):100]) %*% err.all[(init+1):100])/length
        (err.all[(init+1):100])
104 MSE.all[c] <- MSE
105 c <- c + 1
106 }
107
108 plot(lambdas, MSE.all, col="blue",
109      xlab= 'Îť value', ylab='MSE',
110      main='Optimization of Îť')
111
112 lambda_opt <- lambdas[which.min(MSE.all)] # 0.969
113
114 ### Question 4.1 ####
115
116 # Split the data into training and testing sets
117 xtrain <- data$time[1:718]
118 ytrain <- data$co2[1:718]
119 xtest <- data$time[719:738]
120 ytest <- data$co2[719:738]
121
122 # Get the number of observations in training set
123 ntrain <- length(xtrain)
124
125 ## Construct design matrix ##
126
127 # Set the period of the harmonic part
```

```
128 p <- 1 # we've set p to 1 because we can see from the observations
        that the period of the harmonic part is approxiately 1 (1 year
    )
129
130 # Construct the design matrices for the training and testing sets,
        including Fourier terms to capture periodicity in the data
131 Xtrain <- cbind(1, xtrain, sin(2*pi*xtrain/p), cos(2*pi*xtrain/p))
132 Xtest <- cbind(1, xtest, sin(2*pi*xtest/p), cos(2*pi*xtest/p))
133
134 ## Local linear trend model for the data.
135 lambda <- lambda_opt
136 f <- function(j) rbind(1, j, sin(2*pi*j/p), cos(2*pi*j/p))
137 L <- t(matrix(c(1,0,0,0, 1/12,1,0,0, 0,0,cos(2*pi/p/12),sin(2*pi/p
        /12), 0,0,-sin(2*pi/p/12),cos(2*pi/p/12)),ncol=4))
138 LInv <- solve(L)
139
140 # 4 parameters so at 4 observations are needed to estimate theta
141 # however one more is needed to get an estimate of the uncertainty
        .
142
143 init <- 5 # Skip estimating for the first 10 observations
144 ## FNinit & hNinit (First observations) using equation (3.100)
145 F <- matrix(0, nrow=4, ncol=4)
146 h <- matrix(0,nrow=4,ncol=1)
147 for (j in 0:(init-1)){
148   F <- F + lambda^(j) * f(-j/12) %*% t(f(-j/12))
149   h <- h + lambda^(j) * f(-j/12) * ytrain[init-j]
150 }
151
152 ## Allocating space
153 np <- length(h)
154 theta.all <- matrix(NA,ncol=np, nrow=ntrain)
155 sigma.all <- rep(NA, ntrain)
156 sd.err.all <- rep(NA, ntrain)
157 yhat.all <- rep(NA, ntrain)
158 err.all <- rep(NA, ntrain)
159
160 ## Solving at time init
161 theta.hat <- solve(F, h)
162 theta.all[init,] <- theta.hat
163 epsilon <- ytrain[1:init] - cbind(1, (-(init-1):0)/12, sin(2*pi*
        (-(init-1):0)/12/p), cos(2*pi*(-(init-1):0)/12/p)) %*% theta.
        hat
164
```

```
165 T <- 0
166 for (j in 0:(init-1)){
167     T <- T + lambda^(j)
168 }
169
170 Sigma_inv <- diag(init)
171 for (j in 0:(init-1)){
172     Sigma_inv[j,j] <- lambda^((init-1-j))
173 }
174
175 sigma.all[init] <- sqrt(t(epsilon) %*% Sigma_inv %*% epsilon/(T -
      np))
176 sd.err.all[init] <- sigma.all[init] * sqrt(1 + t(f(1/12)) %*%
      solve(F) %*% f(1/12))
177 yhat.all[init+1] <- t(f(1/12)) %*% theta.hat
178 err.all[init+1] <- ytrain[init+1] - yhat.all[init+1]
179
180 ## Looping over the remaining observations
181 for (i in (init+1):(ntrain)){
182   F <- F + lambda^((i-1)) * f(-(i-1)/12) %*% t(f(-(i-1)/12))
183   h <- lambda * LInv %*% h + f(0)*ytrain[i]
184   theta.hat <- solve(F, h)
185   theta.all[i,] <- theta.hat
186
187   ## Adding uncertainty information
188   epsilon <- ytrain[1:i] - cbind(1, (-(i-1):0)/12, sin(2*pi*(-(i
        -1):0)/12/p), cos(2*pi*(-(i-1):0)/12/p)) %*% theta.hat
189   T <- 0
190   for (j in 0:(i-1)){
191     T <- T + lambda^(j)
192   }
193
194   Sigma_inv <- diag(i)
195   for (j in 0:(i-1)){
196     Sigma_inv[j,j] <- lambda^((i-1-j))
197   }
198
199   sigma.all[i] <- sqrt(t(epsilon) %*% Sigma_inv %*% epsilon/(T -
        np))
200
201   ## Estimating s.d. of estimated parameters
202   sd.err.all[i] <- sigma.all[i] * sqrt(1 + t(f(1/12)) %*% solve(F)
        %*% f(1/12))
203
```

```
204   yhat.all[i+1] <- t(f(1/12)) %*% theta.hat
205   err.all[i+1] <- ytrain[i+1] - yhat.all[i+1]
206 }
207
208
209 ## Predictions on test set
210 theta_pred <- theta.all[718,] # Get last theta of the training set
211 sigma_pred <- sigma.all[718] # Get last sigma
212
213 ntest <- length(xtest)
214
215 # Allocate memory
216 yhat_test.all <- rep(NA, ntest)
217 sd.err_test.all <- rep(NA, ntest)
218
219 for (j in 1:ntest){
220       yhat_test.all[j] <- t(f(j/12)) %*% theta_pred # Make
          prediction
221       sd.err_test.all[j] <- sigma_pred * sqrt(1 + t(f(j/12)) %*%
          solve(F) %*% f(j/12)) # Calculate standard deviation of
          the error
222 }
223
224 ## Compute MSE
225 MSE_test <- sum((ytest-yhat_test.all)^2)/ntest
226 MSE_train <- sum((ytrain[10:length(xtrain)]-yhat.all[10:length(
    xtrain)])^2)/ntest
227
228 ## Plot ##
229 t.quan <- qt(p = 0.975, df=rep(ntrain-np, ntrain-9) ) # Get t
    distribution for training
230 t.quan[1:init] <- NA
231 t.quan_test <- qt(p = 0.975, df= rep(ntest-np, ntest)) # Get t
    distribution for testing
232
233 # Plot the training and testing sets
234 plot(xtrain[10:length(xtrain)], ytrain[10:length(xtrain)], col="
    blue",
235       xlab="Time [year]", ylab="Atmospheric CO2 [ppm]",
236       main="Predictions after 2010 (Îť=0.983)",
237       xlim=c(2010, 2020), ylim=c(380, 420))
238 points(xtest, ytest, col="red")
239 matlines(xtrain[10:length(xtrain)], yhat.all[10:length(xtrain)] +
```

```
    t.quan * cbind(0,-sd.err.all[10:length(xtrain)],sd.err.all[10:
    length(xtrain)]),type="l",lty=c(1,2,2),lwd=2, col="darkgreen")
240 matlines(xtest, yhat_test.all + t.quan_test * cbind(0,-sd.err_test
    .all, sd.err_test.all),type="l",lty=c(1,2,2),lwd=2, col="purple
    ")
241 legend("topleft", c("Training set", "Testing set", "Train
    predictions with 95% prediction interval", "Test predictions
    with 95% prediction interval"), col=c("blue", "red", "darkgreen
    ", "purple"), lty=1, bty='n', lwd=2)
```