Technical University of Denmark

Time Series Analysis (02417)

Assignment 1: Atmospheric CO2

Students:

Martin Nguyen, s194378
Asbjørn Hammershøi Petersen, s204084
Lucie Ricq, s230166

February 24, 2023

Contents

1.0: Introduction

In the past decades, the concentration of CO2 in the atmosphere has been steadily increasing, which is a driving factor of climate change. In this assignment, we are aiming to investigate the evolution of atmospheric carbon dioxide (CO_2) concentrations by using different linear models, specifically the Ordinary- and Weighted Least Squares models and a local linear trend model. By using historical data from 1959 to 2017, the goal is to make predictions a couple of years ahead for 2018 and to September 2019. The data set is provided by the National Oceanic and Atmospheric Administration, NOAA/ESRL, and contains monthly observations from Mauna Loa, Hawaii starting from 1959 and ranging to September of 2019 [1]

1.1: Plotting

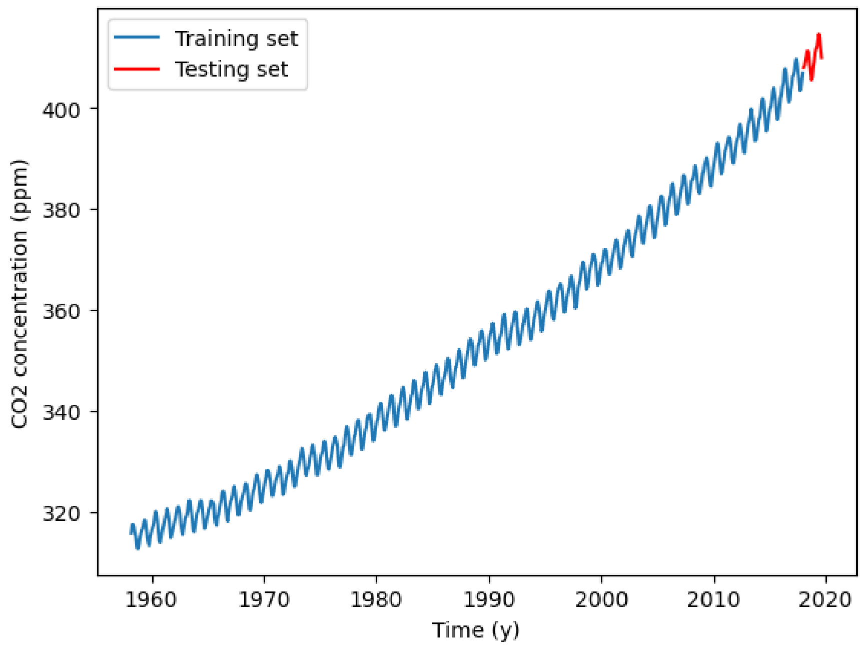

Figure 1: Atmospheric CO_2 concentration measures (1959-2019) with the test set marked from, and including, 2018

The data contains measurements of the atmospheric CO_2 concentration over time in months. The data is partitioned into a training set and a test set where the training set contains data from 1959 to 2017 and the test set from 2018 to 2019 (last 20 observations in the dataset). The test set is only used as a reference for future predictions and will not be used in any training of the different linear models.

Figure 1 shows a global increasing trend in the CO_2 concentration as well as seasonal variations, which is mainly due to the biological processes, i.e. plants through photosynthesis and respiration release CO_2 during fall and winter while absorbing a large amount of CO_2 in the spring and summer[2].

The analytical part of this assignment has been programmed in R, and most of the graphs were done in Python. The code is attached as a .zip file.

1.2: OLS and WLS

For the Ordinary- and Weighted Least Squares models, the following model structure was anticipated to be

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

The model in Equation 1 contains an intercept α and a slope β_t, as well as an annual harmonic part dependent on the parameters β_s, β_c, and the period p. Using a least-squares method, the period was found to be $p \approx 1$, and all parameters were fitted to the data as an initial analysis. For ease of use, this parameter was kept constant throughout this section, with the assumption that it would not change significantly during the analysis. Additionally, some independent noise is assumed to be present in the model.

1.2.1-3

The OLS model is formulated by finding the parameter estimates $\hat{\boldsymbol{\theta}}$ such that they reduce the sum of squared errors between the linear model and the observed data $\mathbf{Y}$. The model is on the form

$$\mathbf{Y} = \mathbf{x}\boldsymbol{\theta} + \boldsymbol{\epsilon} \tag{2}$$

with $\mathbf{x}$ containing the basis functions of the model previously stated. Simply isolating the model for the model errors, and squaring it, grants the following

$$S(\boldsymbol{\theta}) = \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon} = (\mathbf{Y} - \mathbf{x}\boldsymbol{\theta})^T(\mathbf{Y} - \mathbf{x}\boldsymbol{\theta}) \tag{3}$$

Which is differentiated and set equal to 0 as such $\frac{\partial}{\hat{\boldsymbol{\theta}}}S(\hat{\boldsymbol{\theta}}) = 0$. By doing so, the minimum, and thereby the best parameter estimates for this method, can be found by solving what is commonly known as the normal equation, here with the parameter estimates isolated.

$$\mathbf{x}^T\mathbf{x}\hat{\boldsymbol{\theta}} = \mathbf{x}^T\mathbf{Y} \rightarrow \hat{\boldsymbol{\theta}} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{Y} \tag{4}$$

The uncertainty of the parameter estimates can be measured by finding the variance or standard deviation of them. Finding the variance can be done by

$$V[\hat{\boldsymbol{\theta}}] = \sigma^2(\mathbf{x}^T\mathbf{x})^{-1} \tag{5}$$

with the estimated variance of the residuals, found by

$$\hat{\sigma}^2 = \frac{\boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon}}{n - p} \tag{6}$$

with n being the number of data points and p here being the number of parameters, in this case, 4.

Using these methods, fitting the data using the train split provides with the following parameter estimates, and the following error on the test set. Looking at the parameter estimates, it

$\boldsymbol{\theta}$	$\hat{\boldsymbol{\theta}}$	$V[\hat{\boldsymbol{\theta}}]$
α	-2710	225
β_t	1.54	5.6810^{-5}
β_s	2.61	0.0339
β_c	-1.05	0.0340
Test MSE	86.31	

Table 1: The parameter estimates and their variance

is clear that they are quite certain, which is to be expected, as the model seems to overall fit the data quite well. The variance of the parameter estimates is quite small, which indicates low uncertainty. However, as it is tested on the test set, it's clear that the model has a very high bias toward the training data. It is therefore not very good at generalizing to the non-linear trend that seems to be present in the data, which can be seen in Figure 2. To help mitigate this a Weighted Least Squares model can be used.

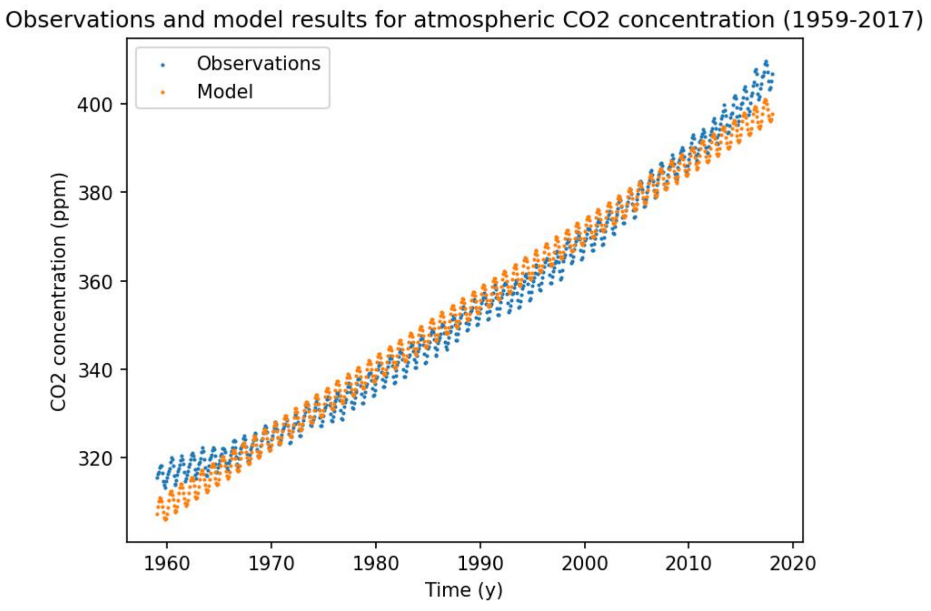

Figure 2: The estimated OLS model plotted with the observed values

As a side note, the Maximum Likelihood estimators are not used, since the observations, and thereby the residuals, does not seem to be normally distributed.

1.2.4-8

The WLS model structure is the same as in the OLS model, but the parameter estimates will of course be slightly different. There are many reasons to assign different weights to observations. For example, it could be that newer observations are more important than older ones, which could be the case here. To estimate the parameters, a covariance matrix Σ of the residuals is introduced, which can take on many different structures.

The estimates are obtained by solving the normal equation and isolating $\hat{\boldsymbol{\theta}}$:

$$\hat{\boldsymbol{\theta}} = (\mathbf{x}\Sigma^{-1}\mathbf{x})^{-1}\mathbf{x}^T\Sigma^{-1}\mathbf{Y}, \tag{7}$$

assuming that $\mathbf{x}^T\Sigma\mathbf{x}$ has full rank; otherwise, it is not possible to isolate the parameter estimates in the normal equation. For this particular data set, the correlation structure Σ is assumed to follow an exponential decay, such that

$$\mathrm{Cor}(\varepsilon_{t_i}, \varepsilon_{t_j}) = \rho^{|t_i - t_j|}$$

Yielding the following covariance matrix structure, which is used to weigh the latest observation more than the early observations.

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

Firstly, the relaxation algorithm is used to estimate Σ. To perform this algorithm, a covariance structure is first selected, and then the matrix is estimated. The parameter estimates are then calculated, and the residuals of the resulting model are used to estimate Σ again. This process can be repeated multiple times until convergence, as the normal equation is continually solved to minimize the squared error using each new iteration of Σ. In this project, ρ is first estimated by taking the autocorrelation of the residuals from the OLS model. Autocorrelation here refers to taking the correlation between each adjacent residual by creating two new vectors from every residual except the first and except the last, respectively. The first value of ρ is $Cor(\varepsilon_{[2:N]}, \varepsilon_{[1:N-1]}) = 0.982087$, and it is passed into Σ. Using Equation 7, estimates are then found, and these estimates are used to generate new residuals, which are again used for their autocorrelation. The change in ρ is minuscule in this case, as are the changes in the parameter estimates. The variance of the WLS estimators is found as follows:

$$V[\hat{\boldsymbol{\theta}}] = \sigma^2(\mathbf{x}^T\Sigma^{-1}\mathbf{x})^{-1} \tag{9}$$

and the estimator for σ is found, much like for the OLS estimator:

$$\hat{\sigma}^2 = \frac{(\mathbf{Y} - \mathbf{x}\hat{\boldsymbol{\theta}})^T \Sigma^{-1} (\mathbf{Y} - \mathbf{x}\hat{\boldsymbol{\theta}})}{N - p} \tag{10}$$

Finally, using the algorithm the estimated WLS parameters are obtained which can be seen in Table 2

Iteration	1	2	3	...
$\boldsymbol{\theta}$	$\hat{\boldsymbol{\theta}}_1$	$\hat{\boldsymbol{\theta}}_2$	$\hat{\boldsymbol{\theta}}_3$	$V[\hat{\boldsymbol{\theta}}_3]$
α	-2743.7555	-2744.0263	-2744.0296	16535
β_t	1.5581	1.5583	1.5583	0.00418
β_s	2.6477	2.6477	2.6477	0.00458
β_c	-1.0188	-1.0188	-1.0188	0.00456

Table 2: Convergence of the relaxation algorithm after 3 iterations

The biggest change of the model, from the OLS estimates, happens in the very first iteration upon the introduction of Σ which seems to nudge the slope up and make the amplitude of the harmonic part slightly bigger, which is indeed what could improve the model if ever so slightly.

Using the parameter estimates from when there is convergence, the test set can again be used to measure performance. Upon taking the Mean Squared Error, the WLS model obtains an MSE of 59, which is an improvement from the OLS model. Comparing Figure 2 and Figure 3, it can be seen that the entire slope has been nudged up just a bit, which is likely what contributes the most to the improvement. Obviously, this still does not fix the issue of a non-linear trend in the data, which makes both models fairly weak in prediction. However, the WLS model fares better and could perhaps be initialized even better with a different approach. However, upon examining the variance of the WLS parameter estimates, it can be seen that they are quite a bit higher than for the OLS model, which indicates a higher uncertainty of the estimates and as a result, higher uncertainty in the predictions, using the model.

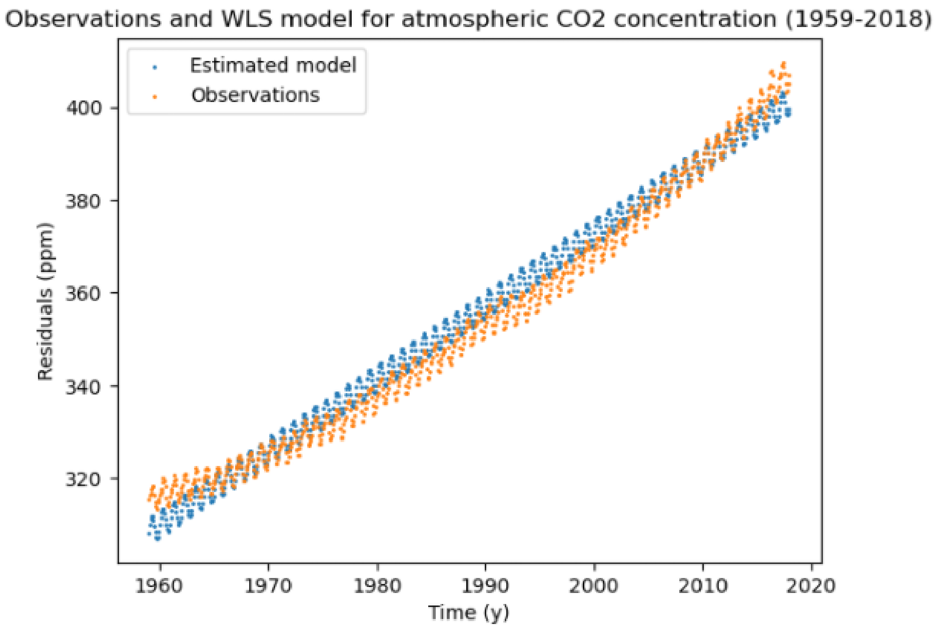

Figure 3: The estimated WLS model against the observed values

The OLS- and WLS model can be seen together with the actual observations in Figure 4. It can be seen that the WLS is slightly closer to the observations after 2018.

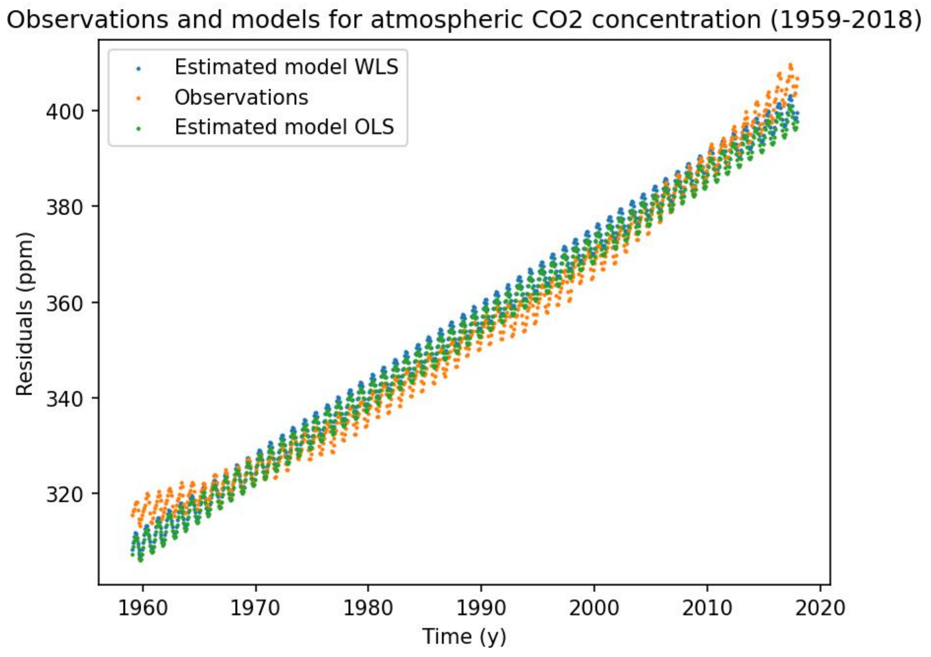

Figure 4: Comparison between OLS and WLS models

1.3: Local linear trend

In this section, the available information is limited to observations of the dependent variable $\mathbf{Y}$. Therefore, any modeling or prediction in this context will rely solely on the observed values of $\mathbf{Y}$. To achieve this, trend models are introduced, which are regression models that consider specific time functions as independent variables. The general form of a trend model is given as:

$$Y_{N+j} = \mathbf{f}(j)^T \theta_N + \epsilon_{N+j} \tag{11}$$

where $\mathbf{f}(j) = [f_1(j), ..., f_p(j)]^T$ is a vector of known forecast functions, θ is a vector of parameters and ϵ is the noise component with the same assumptions as mentioned earlier. Additionally, the forecast function satisfies the difference equation

$$\mathbf{f}(j+1) = \mathbf{L}\mathbf{f}(\mathbf{j}), \quad [\text{start value } \mathbf{f}(0)] \tag{12}$$

where the transition matrix $\mathbf{L}$, is a fixed $p \times p$ non-singular matrix.

1.3.1

It is only reasonable to assume the trend model that corresponds to the linear model in the previous question resembles a harmonic model with the period p. Therefore, the trend model written in the form of Equation 11 becomes

$$Y_{N+j} = \theta_0 + \theta_1 j + \theta_2 \sin(\frac{2\pi}{p} j) + \theta_3 \cos(\frac{2\pi}{p} j) \tag{13}$$

This form is obtained when the vector of forecast functions, $\mathbf{f}(j)$, is

$$\mathbf{f}(j) = [1, j, \sin(\frac{2\pi}{p} j), \cos(\frac{2\pi}{p} j)]^T \tag{14}$$

and the transition matrix, $\mathbf{L}$ and forecast functions, $\mathbf{f}(j)$, evaluated at $j = 0$ are therefore

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & \cos(\frac{2\pi}{p}) & \sin(\frac{2\pi}{p}) \\ 0 & 0 & -\sin(\frac{2\pi}{p}) & \cos(\frac{2\pi}{p}) \end{bmatrix}, \quad \mathbf{f}(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{15}$$

Defining the time unit of the forecasting functions, j, becomes critical since only the observations of the dependent variable, $\mathbf{Y}_N$, are available. The time unit used is based on months, and so the period of the harmonic part becomes $p = 12$, instead of the estimated $p \approx 1$ from the earlier when estimating the linear model using OLS and WLS (where also the independent variable $\mathbf{x}$ was available).

1.3.2

Considering the first 10 observations as transient, the formulas shown below for calculating the matrix and vector, $\mathbf{F}_{10}$ and $\mathbf{h}_{10}$, respectively, are used to estimate $\hat{\theta}_{10}$.

$$\begin{aligned} \mathbf{F}_N &= \Sigma_{j=0}^{N-1} \lambda^j f(-j) f(-j)^T \\ \mathbf{h}_N &= \Sigma_{j=0}^{N-1} f(-j) Y_{N-j} \\ \hat{\theta}_N &= F_N^{-1} h_N \end{aligned} \tag{16}$$

The following observations are now evaluated iteratively meaning $\mathbf{F}_{N+1}, \mathbf{h}_{N+1}$ and $\hat{\theta}_{N+1}$ in an adaptive scheme become updated as the next observation is considered. However, $\mathbf{F}_{N+1}$ and $\mathbf{h}_{N+1}$ are updated after the one-step prediction $\hat{Y}_{N+1|N}$ such that the computation should be done in the given order

$$\begin{aligned} \hat{Y}_{N+1|N} &= \mathbf{f}^T(1)\hat{\theta}_N \\ \mathbf{F}_{N+1} &= \mathbf{F}_N + \lambda^N \mathbf{f}(-N)\mathbf{f}^T(-N) \\ \mathbf{h}_{N+1} &= \mathbf{L}^{-1}\mathbf{h}_N + \mathbf{f}(0)Y_{N+1} \\ \hat{\theta}_{N+1} &= \mathbf{F}_{N+1}^{-1}\mathbf{h}_{N+1} \end{aligned} \tag{17}$$

The recursive procedure in Equation 17 is repeated until the last observation (Y_{718}) in the training set. Figure 5 shows the filtering of the historical data with the actual observations included.

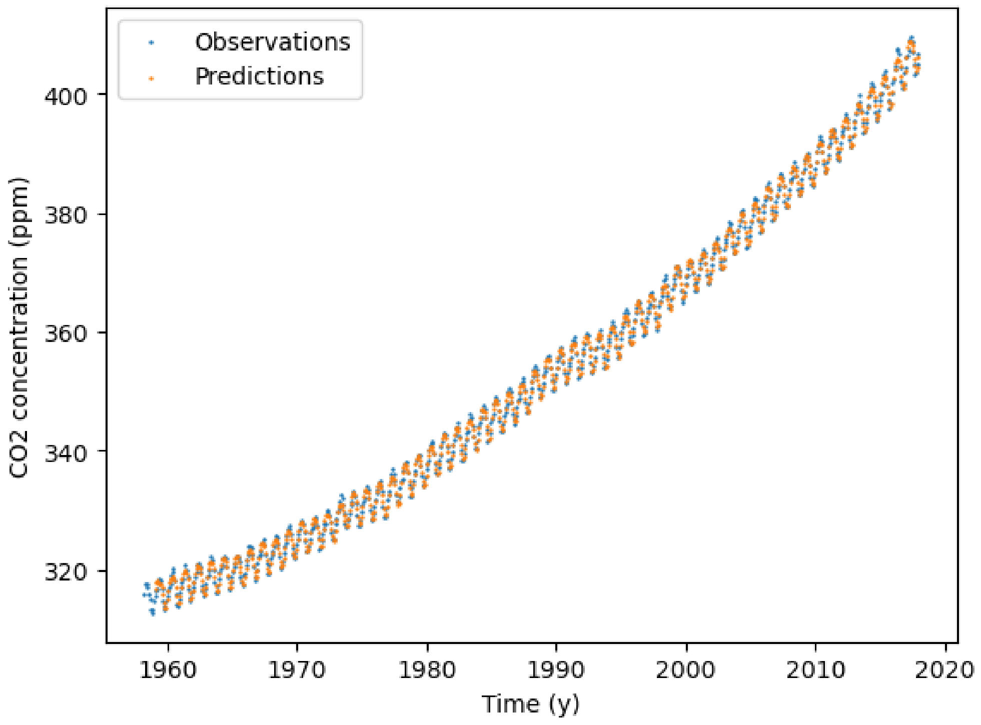

Figure 5: Filtering the historical data

The local trend model encapsulates the behavior and time evolution of the atmospheric CO_2 concentration more accurately than what the OLS- and WLS estimates could as seen in Figure 4. This shows that the adaptive update of the estimates corresponds to the local trend that happens in the span of twelve months which includes a slightly increasing intercept as well as the harmonic variation.

1.3.3

As mentioned earlier, the 10 first observations were considered transient, which means the first estimate of the parameters was computed for $\hat{\theta}_{10}$. Therefore, the first prediction error was calculated for $\varepsilon_{11} = Y_{11} - \hat{Y}_{11|10}$.

The general form for the prediction error, ε_{N+1}, is given as

$$\varepsilon_{N+1} = Y_{N+1} - \hat{Y}_{N+1|N} \tag{18}$$

A local estimate of σ_N^2 based on WLS is given by

$$\hat{\sigma}_N^2 = \frac{1}{T-p}(\mathbf{Y} - \mathbf{x}_N\hat{\theta}_N)^T\Sigma^{-1}(\mathbf{Y} - \mathbf{x}_N\hat{\theta}_N), \quad T = \Sigma_{j=0}^{N-1}\lambda^j \tag{19}$$

Figure 6 shows plots of the one-step prediction errors, ε_{N+1}, as well as the local estimates of the variances, $\hat{\sigma}_N^2$.

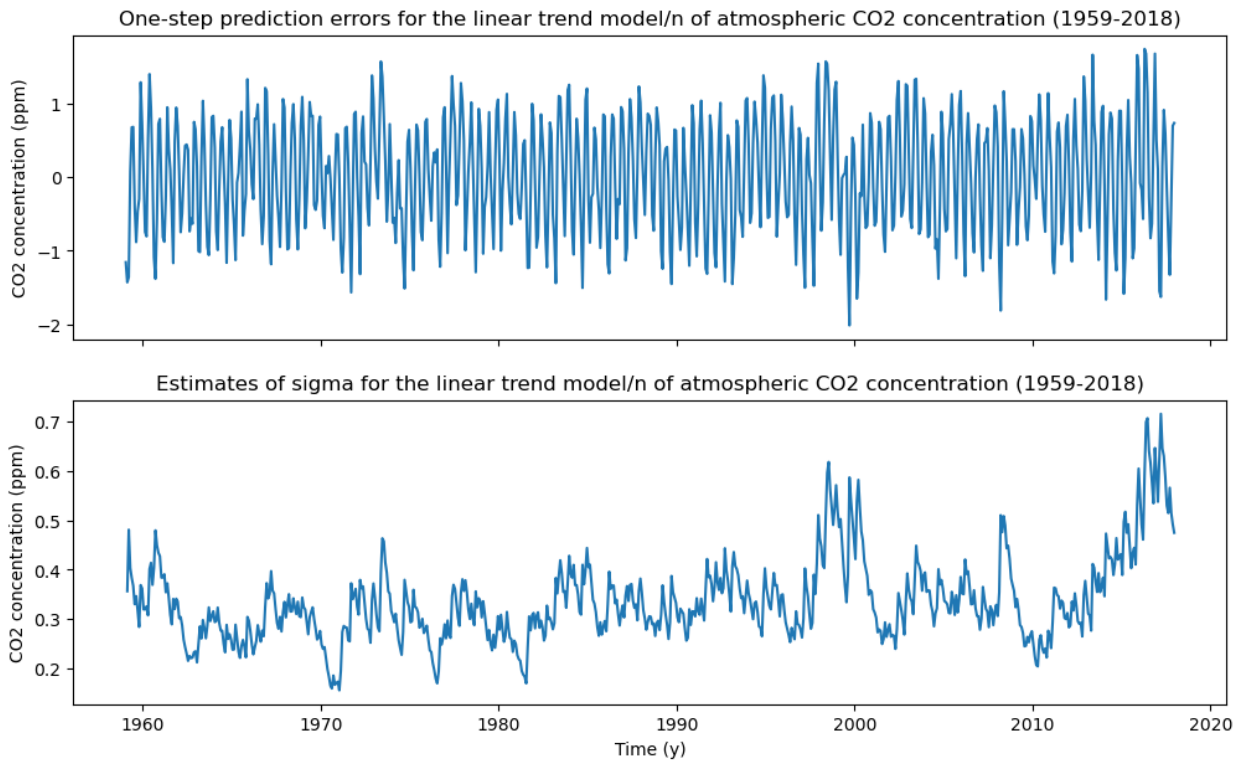

Figure 6: Residuals and the local estimate of the variance on the training data

It can be seen that the one-step prediction errors on Figure 6 are very restricted, most of them being in the interval $[-1, 1]$. Seemingly, the estimates of $\hat{\sigma}_N^2$ range between 0.2 and 0.7. It will be investigated in the following questions how this impacts the prediction interval since $\hat{\sigma}_N$ involves the formulation of the bounds of the prediction interval.

1.3.4-5

The 95% one-step prediction interval is computed using the following formula

$$\hat{Y}_{N+1|N} \pm t_{0.05}(N-p)\hat{\sigma}\sqrt{1+\mathbf{f}^T(1)\mathbf{F}_N^{-1}\mathbf{f}(1)} \tag{20}$$

When looking in Figure 7 the 95% prediction intervals over the whole time evolution seem quite small, which may suggest that the model is fitted fairly accurately, and with a low variance of the predictions. Fortunately, most of the observations actually lie in between the bounds of the prediction interval.

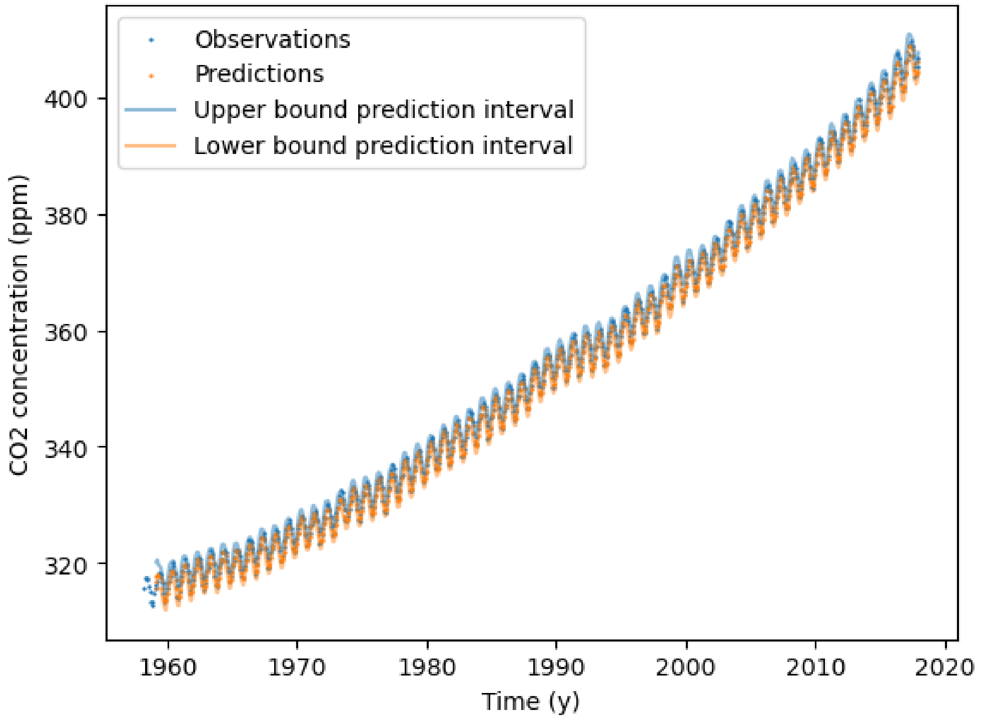

Figure 7: Prediction interval of the local trend model

Zooming in on the years from 2010 to 2018 as shown in Figure 8, it can also be observed that the actual observation in this time period lies within the prediction intervals. This reassures that the linear trend model is actually fitted fairly accurately.

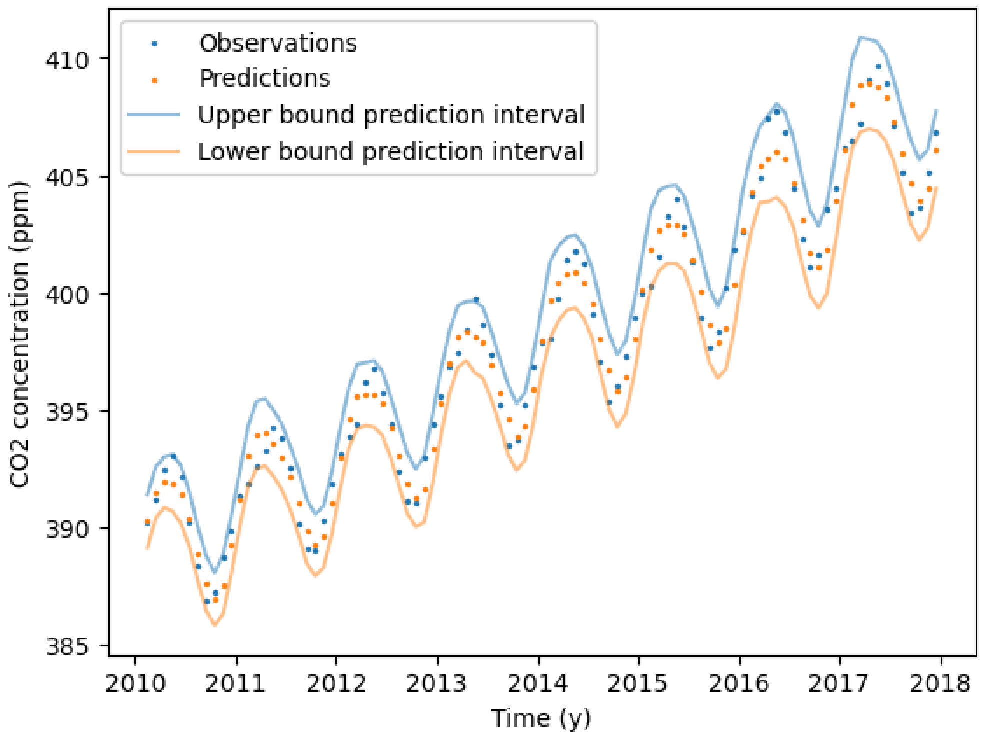

Figure 8: Detail on the last years of the training data (2010-2018)

1.3.6-7

For this section, forecasts for different months ahead using the last estimated parameters, $\hat{\theta}_{718}$, from the training set will be performed. The forecasted months (1, 2, 6, 12, and 20 months ahead) will be compared with the test set (from 2018 to September 2019) that is used as a reference. These different forecasts are calculated using the following formula

$$\hat{Y}_{718+l|718} = \mathbf{f}^T(l)\hat{\theta}_{718} \tag{21}$$

Table 3 shows the forecasted values paired against the corresponding value in the test set. The values only differ by small decimal points which indicates that the latest estimate of the parameters, $\hat{\theta}_{718}$, are quite representative for the forecasting of 20 months ahead. This means that the trend doesn't change significantly in the following years after the last observation in the training set.

Months ahead	**1**	**2**	**6**	**12**	**20**
Predicted CO2 concentration (ppm)	408.21	410.08	410.83	409.00	410.64
Actual CO2 concentration (ppm)	407.96	408.32	410.79	409.07	409.95

Table 3: *1*-, *2*-, *6*-, *12*-, and *20*-step prediction of the test data using the local linear trend model with the forgetting factor, $\lambda = 0.90$

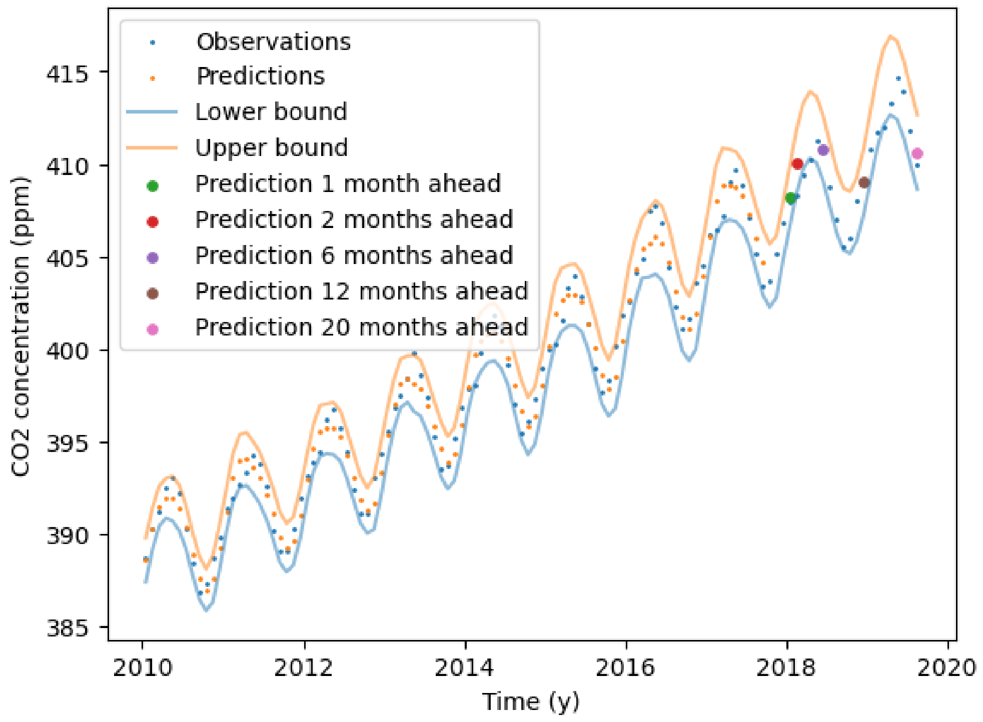

Figure 9: The forecasts using the last estimate of theta using the training set

Figure 9 visually shows that the forecasted months ahead are well-fitted to the actual observations. Indeed, they are effectively following the harmonic trend in the data. That said, it can be observed that the amplitude of the harmonic variation is slightly too low to match the observations completely. However, it is unexpected that the prediction intervals do not seem to broaden out after 2018, but it is suspected that the local trend simply fits quite well on the test data.

1.3.8

On Figure 10 the slope of each $\hat{\theta}$ has been used, as it is should be representative to the estimated value of that time step, in accordance to the linear trend model. The mean value is, as expected, not entirely linear, and is constantly nudged due to the harmonic nature of the observations. It is quite accurate, which is further testified from the previous results, as the harmonic trend is grown out from this baseline.

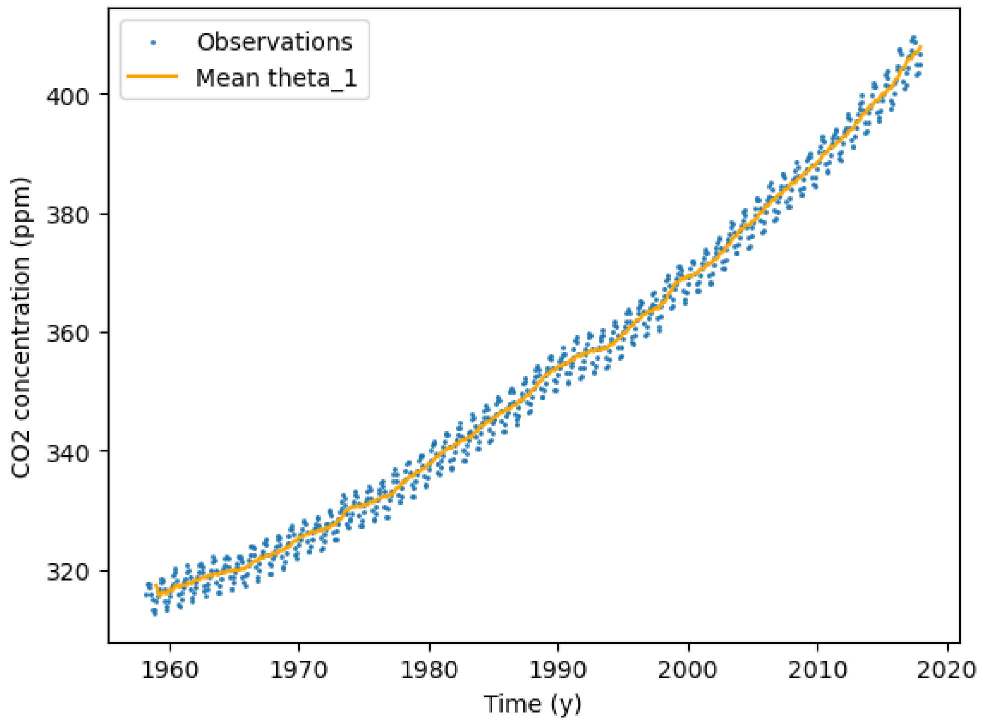

Figure 10: Observations and 1-step predictions for the whole period (1959-2019)

1.4: Optimal λ

The forgetting factor λ determines how much each observation should be weighted, and in order to find the optimal λ the one-step prediction error, ε_t, and the sum of squared one-step prediction errors, $S(\lambda)$, should be considered shown below.

$$\varepsilon(\lambda) = Y_t - \hat{Y}_{t|t-1}(\lambda) \tag{22}$$

The computation of Equation 22 happens in the recursive updating as shown in Equation 11 where the prediction at the given time step is calculated. However, instead of 10 observations, 100 observations are now considered as the burn-in period or "transient". The sum of squared one-step prediction errors (SSE) is then

$$S(\lambda) = \Sigma_{t=1}^{N} \varepsilon_t^2(\lambda) \tag{23}$$

The constant λ that minimizes $S(\lambda)$ is used, and various techniques could be done for the minimization such as gradient descent. In this report, the minimization is evaluated using grid-search; SSE of the one-step predictions are calculated for various λ values ranging from 0.01 to 1.

A plot of the sum of squared prediction errors (SSE) against the different values are shown in Figure 11.

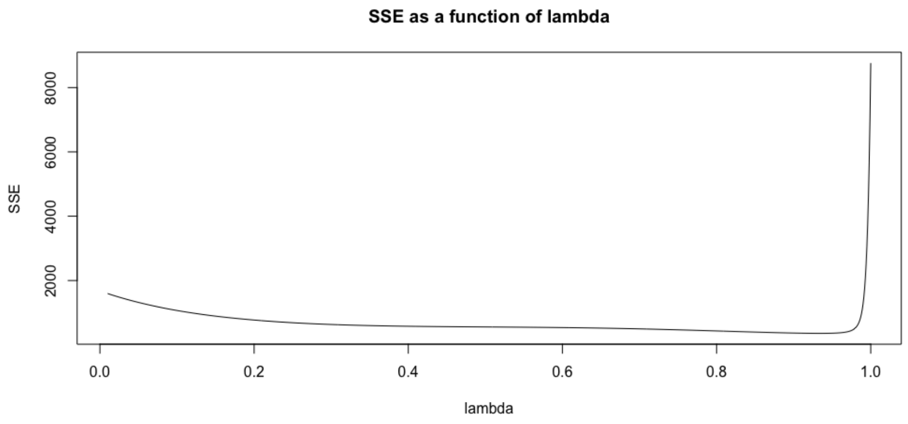

Figure 11: SSE, $S(\lambda)$, as a function of the forgetting factor, λ.

Figure 11 shows that the SSE steadily decreases as λ increases from 0.01 to 0.93 and increases significantly onward to 1. The minimum SSE, $S(\lambda) = 356.2$, was evaluated to be for $\lambda = 0.93$.

1.4.1

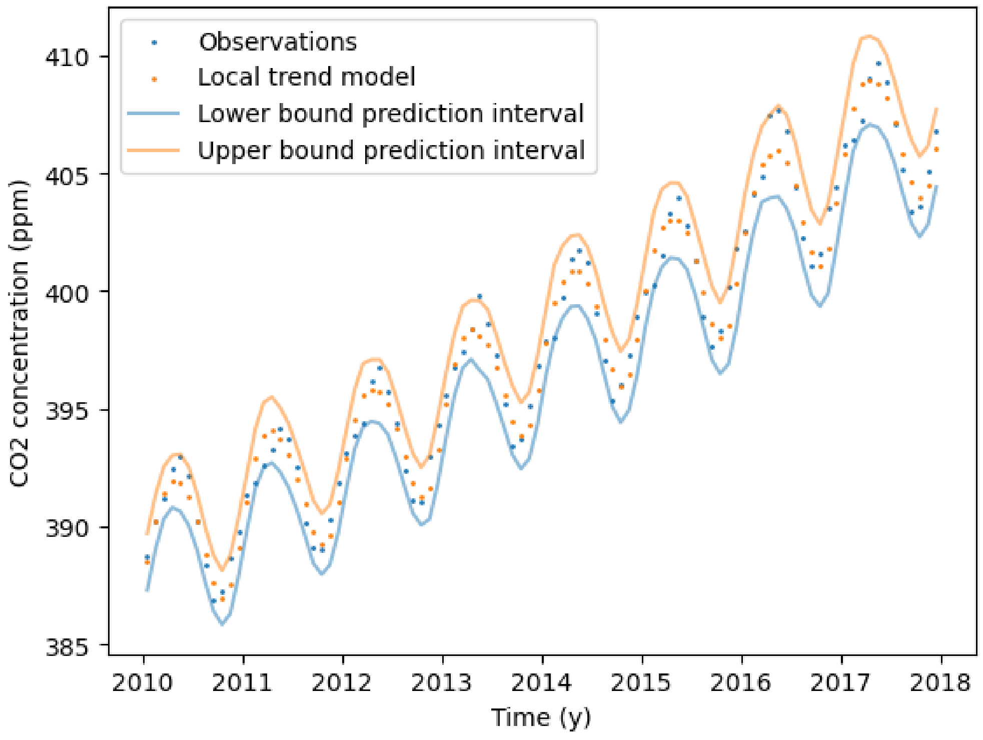

Figure 12: Actual observations and local trend model prediction for $\lambda = 0.93$ (2010-2017)

Figure 12 shows the actual observations and predicted values based on the local linear trend model with the optimal forgetting factor, $\lambda = 0.93$. The 95% prediction interval is also included. The time axis is zoomed closely into the years between 2010 and 2017 and from here it can be seen that the one-step predictions are estimated to be rather close to the actual observations. Additionally, the prediction interval seems to be quite consistent during the whole time evolution between 2010 to 2017.

1.4.2

Months ahead	**1**	**2**	**6**	**12**	**20**
Predicted CO2 concentration (ppm)	408.06	409.90	410.74	408.83	410.54
Actual CO2 concentration (ppm)	407.96	408.32	410.79	409.07	409.95

Table 4: *1*-, *2*-, *6*-, *12*-, and *20*-step prediction of the test data using the local linear trend model with $\lambda = 0.93$

Table 4 shows the predicted CO2 concentrations for 1, 2, 6, 12-, and 20 months ahead in comparison with the actual predicted CO2 values in the given months after 2017. It can be observed from the table that the linear trend model with the forgetting factor, $\lambda = 0.93$, predicts closely to the actual observed values. This does not come as a surprise as we witnessed earlier in Table 3 when $\lambda = 0.90$.

1.4.3

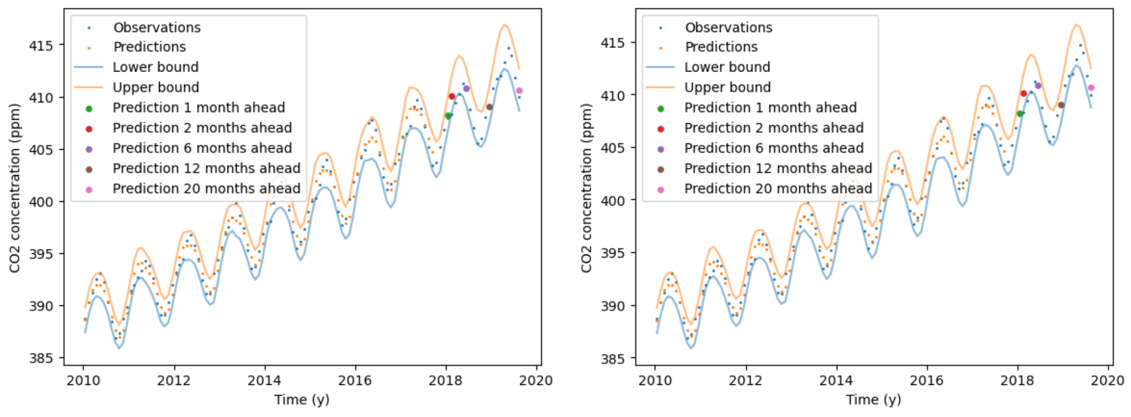

Figure 13: Plots of the actual observations and predictions of the local trend model for $\lambda = 0.90$ (left) and $\lambda = 0.93$ (right) with 95 % prediction interval.

Figure 13 shows the local linear trend models using the initial forgetting factor, $\lambda = 0.90$ (left), and the optimal forgetting factor, $\lambda = 0.93$ (right). As the optimal λ is only 0.03 higher than the λ used in the earlier analysis, it can be seen that the predictions and thereby the local linear trend models with these two different forgetting factors do not differ significantly from each other. This is also confirmed when looking at Table 5, which shows that the predictions using the two different forgetting factors are almost the same.

Months ahead	**1**	**2**	**6**	**12**	**20**
Predicted CO2 concentration (ppm, $\lambda = 0.90$)	408.21	410.08	410.83	409.00	410.64
Predicted CO2 concentration (ppm. $\lambda = 0.93$)	408.06	409.90	410.74	408.83	410.54
Actual CO2 concentration (ppm)	407.96	408.32	410.79	409.07	409.95

Table 5: *1*-, *2*-, *6*-, *12*-, and *20*-step prediction comparison of $\lambda = 0.90$, $\lambda = 0.93$, and the actual observations

1.5: Overall

In regard to the different models used in this project, the local trend model stands out the most as it manages to account for the slight quadratic or perhaps exponential shape of the observations, and thereby shows the ability to predict with decent accuracy. The OLS- and WLS models perform quite poorly in comparison, and many ways and are not very future-proof. Interesting future avenues would be to investigate using a different model such as a quadratic or exponential incorporated with OLS or WLS. This would make them able to better follow the overall trend of the data, however with the disadvantage that they would likely start heavily overestimating the future, instead of underestimating it.

References

[1] Thoning, “Monthly average mauna loa co2.”

[2] G. Shirah, “Seasonal changes in carbon dioxide.”

Appendix

Much of the plotting code has been omitted. The written CSV files in the code, are purely done for plotting in Python.

1.2

```
# co2_data <− read.table("A1_co2.txt", header=TRUE)
# split <− 2016
# datasplit <− co2_data[co2_data$time≤split,]

data <− read.csv("../data.csv", sep=" ")
# data_matrix <− as.dataframe(data)
data_train <− data[1:718,1:4]
# x_train <− data_matrix[1:718,3]

## 1.2.1
model <− lm(co2 ¬ 1 + time + I(sin((2∗pi/1)∗time)) + ...
    I(cos((2∗pi/1)∗time)), data=data_train)
summary(model)

## 1.2.5
n <− length(data_train$time)
rho <− 0.5

build_cor <− function(n, rho){
  Sigmay <− vector("list", n)
  for (i in 1:n) {
    rows <− to_vec(for(k in 1:n) ifelse(k≤i, 0, rho^(k−i)))
    Sigmay <− rbind(Sigmay, rows)
  }
  Sigmay <− Sigmay[−1,]

  diag <− rep(1,n)
  diag(Sigmay) <− diag
  Sigmay[lower.tri(Sigmay)] <- t(Sigmay)[lower.tri(Sigmay)]
  Sigmay <- as.numeric(Sigmay)
  Sigmay <- matrix(Sigmay, nrow=n, ncol=n)
  return(Sigmay)

}

Sigmay <- build_cor(n, rho)

# 1.2.5 - THE RELAXATION ALGORITHM
x <- data_train$time
X <- cbind(1,x,sin(2*pi*x),cos(2*pi*x))
# x_new <- news$time
# X_new <- cbind(1,x_new,sin(2*pi*x_new),cos(2*pi*x_new))
y <- data_train$co2
resids <- residuals(model)
rho <- cor(resids[-1], resids[-length(resids)])
thets <- c()
rho <- 1
for(i in 1:5){
  thetahat <- ...
     solve(t(X)%*%solve(Sigmay)%*%X)%*%(t(X)%*%solve(Sigmay)%*%y)
  eps <- y - X%*%thetahat
  rho <- cor(eps[-1], eps[-length(eps)])
  Sigmay <- build_cor(n, rho)
  print(i)
  thets <- cbind(thets, thetahat)
}
# WLS_preds <- append(X%*%thetahat, X_new%*%thetahat)
```

1.3

```
    ## Question 1.3.2
data <- read.csv("../data.csv", sep=" ")
data_matrix <- as.matrix(data)
matrix_train <- data_matrix[1:718,1:4]
x_train <- data_matrix[1:718,3]
n <- length(x_train)
y_train <- matrix(data_matrix[1:718,4],nrow=n,ncol=1)
p <- 4
lambda <- 0.9
```

```
p_harm <- 12
f <- function(j) rbind(1,j,sin(2*pi/p_harm*j),cos(2*pi/p_harm*j))
L <- t(matrix(c(1,0,0,0, 1,1,0,0, ...
    0,0,cos(2*pi/p_harm),sin(2*pi/p_harm), ...
    0,0,-sin(2*pi/p_harm),cos(2*pi/p_harm)),ncol=4))
LInv <- solve(L)


init <- 10
F <- matrix(0, nrow=p, ncol=p)
h <- matrix(0, nrow=p, ncol=1)
for (j in 0:(init-1)){
  F <- F + lambda^j*f(-j)%*%t(f(-j))
  h <- h + lambda^j*f(-j)*y_train[init-j]
}

## Allocating space
theta.all <- matrix(NA,ncol=p, nrow=n)
sigma.all <- rep(NA, n)
sigma.local <- rep(NA, n)
sd.theta.all <- matrix(NA,ncol=p, nrow=n)
epsilon.all <- matrix(NA,nrow=n)
## Solving at time init
theta.hat <- solve(F)%*%h
theta.all[init,] <- solve(F)%*%h
epsilon <- y_train[1:init] - cbind(1, -(init-1):0, ...
    (-sin(2*pi/p_harm*(init-1):0)), -cos(2*pi/p_harm*(init-1):0)) %*% ...
    theta.hat
sigma.all[init] <- sqrt(sum(epsilon^2)/(init-p))
sd.theta.all[init,] <- sigma.all[init] * sqrt(diag(solve(F)))
y_est <- matrix(NA,nrow=n+20,ncol=1)
y_pred_up <- matrix(NA, nrow=n+20, ncol=1)
y_pred_down <- matrix(NA, nrow=n+20, ncol=1)

## Looping over the remaining observations
begin <- init+1
for (i in begin:n){
  y_est[i] <- t(f(1)) %*% theta.hat
  F <- F + lambda^(i-1) * f(-i+1) %*% t(f(-i+1))
  h <- lambda* LInv %*% h + f(0)%*%y_train[i]
  theta.hat <- solve(F)%*%h
  theta.all[i,] <- theta.hat
  epsilon <- y_train[1:i] - cbind(1, -(i-1):0, ...
     (-sin(2*pi/p_harm*(i-1):0)), -cos(2*pi/p_harm*(i-1):0)) %*% ...
     theta.hat
  epsilon_for_sigma <- y_train[i] - y_est[i]
  epsilon.all[i] <- epsilon_for_sigma
  espilon_continu <- y_train[begin:i] - y_est[begin:i]
  sigma <- sqrt((epsilon_for_sigma^2)/(i - p))
  sigma.all[i] <- sqrt(sum(epsilon^2)/(i - p))
  sd.theta.all[i,] <- sigma.all[i] * sqrt(diag(solve(F)))
  if(i>11){
    Sigma_for_pred <- ...
       t(espilon_continu)%*%diag(lambda^((i-1):init))%*% ...
    (espilon_continu)/(sum(lambda^((i-1):0))-p)
    sigma.local[i] <- Sigma_for_pred
    y_pred_up[i] <- y_est[i] + qt(p=0.975, ...
       i-1-p)*sqrt(Sigma_for_pred*(1+t(f(1))%*%solve(F)%*%f(1)))
    y_pred_down[i] <- y_est[i] - qt(p=0.975, ...
       i-1-p)*sqrt(Sigma_for_pred*(1+t(f(1))%*%solve(F)%*%f(1)))
  }
}

## Question 1.3.3
write.csv(epsilon.all,"../created_data/1.3.3_epsilon.all.csv")
write.csv(sigma.all,"../created_data/1.3.3_sigma.all.csv")
write.csv(sigma.local,"../created_data/1.3.3_sigma.local.csv")
epsilon.all

## Question 1.3.4
write.csv(y_est,"../created_data/1.3.4_y_est_lambda_0.9.csv")
write.csv(y_pred_down,"../created_data/1.3.4_pred_down_lambda_0.9.csv")
write.csv(y_pred_up,"../created_data/1.3.4_pred_down_lambda_0.9.csv")

## Question 1.3.6
y_pred1 <- t(f(1)) %*% theta.hat
y_pred2 <- t(f(2)) %*% theta.hat
y_pred6 <- t(f(6)) %*% theta.hat
y_pred12 <- t(f(12)) %*% theta.hat
y_pred20 <- t(f(20)) %*% theta.hat

## Question 1.3.8
for(i in 718:738){
  step = i-718
  y_est[i] <- t(f(step)) %*% theta.hat
  y_pred_up[i] <- y_est[i] + qt(p=0.975, ...
     i-1-p)*sqrt(Sigma_for_pred*(1+t(f(step))%*%solve(F)%*%f(step)))
  y_pred_down[i] <- y_est[i] - qt(p=0.975, ...
     i-1-p)*sqrt(Sigma_for_pred*(1+t(f(step))%*%solve(F)%*%f(step)))
}

write.csv(y_est,"../created_data/1.3.8__y_est_lambda_0.9.csv")
write.csv(y_pred_down,"../created_data/1.3.8__pred_down_lambda_0.9.csv")
write.csv(y_pred_up,"../created_data/1.3.8__pred_down_lambda_0.9.csv")
```

1.4

```
    #### 1.4 ####
# Optimal Lambda
# In this question you should find the optimal lambda
# (minimizing squared one step prediction errors) with a burning
# period of 100 months.

# 1.4.1 # plot with optimal lambda showing the data and predictions ...
   from 2010
# and onwards

## loading data and defining variables
data <- read.table("A1_co2.txt", header=TRUE)
data_matrix <- as.matrix(data)

x <- data_matrix[1:738,3]
Y <- data_matrix[1:738,4]
X <- cbind(1,x,sin(2*pi*x),cos(2*pi*x))

matrix_train <- data_matrix[1:718,1:4]
x_train <- data_matrix[1:718,3] # burnin point of 100 months
y_train <- data_matrix[1:718,4] # burnin point of 100 months

## constants
l_step <- 1
n <- length(y_train) # number of observations
init <- 100 #  transient / burnin-point
p = 4 # number of parameters
lambda_list <- seq(from = 0.01, to = 1, by = 0.001) # list of ...
   different lambda
```

```r

# define L matrix and f vector
p_harm <- 12
f <- function(j) rbind(1,j,sin(2*pi/p_harm*j),cos(2*pi/p_harm*j))
L <- t(matrix(c(1,0,0,0, 1,1,0,0, ...
    0,0,cos(2*pi/p_harm),sin(2*pi/p_harm), ...
    0,0,-sin(2*pi/p_harm),cos(2*pi/p_harm)),ncol=4))
LInv <- solve(L)

epsilon_list <- c()

# loop over all lambda

for (g in 1:length(lambda_list)){
  print(g) # print iteration
  lambda <- lambda_list[g] # calculate with lambda[g]

  # FNinit & hNinit (burning point)
  F <- matrix(0, nrow=p, ncol=p)
  h <- matrix(0,nrow=p,ncol=1)

  for (j in 0:(init-1)){
    F <- F + lambda^j*f(-j)%*%t(f(-j))
    h <- h + lambda^j*f(-j)*y_train[init-j]
  }

  # allocating space
  theta.all <- matrix(NA,ncol=p, nrow=n)
  epsilon.all <- matrix(NA,nrow=n, ncol=1)
  sum_sqr_epsilon <- c()
  y_est <- matrix(NA,nrow=n,ncol=1)


  # solving at time init (100)
  theta.hat <- solve(F)%*%h
  theta.all[init,] <- theta.hat


  for (i in seq(from = init+1, to = n, by = l_step)){
    y_est[i] <- t(f(1)) %*% theta.hat
    F <- F + lambda^(i-1) * f(-i+1) %*% t(f(-i+1))
    h <- lambda* LInv %*% h + f(0)%*%y_train[i]

    theta.hat <- solve(F)%*%h
    theta.all[i,] <- theta.hat

    residual <- y_train[i] - y_est[i]
    epsilon.all[i] <- residual

  }
  skip <- init+1
  sum_sqr_epsilon <- sum(epsilon.all[skip:n]^2) # sum of squarred error
  epsilon_list <- cbind(epsilon_list, sum_sqr_epsilon)

}
# comment on the results
plot(lambda_list, epsilon_list, type = 'l', xlim = c(0.01, 1), ylab = ...
   'SSE', xlab = 'lambda', main='SSE as a function of lambda')

min(epsilon_list)
idx <- which.min(epsilon_list)
lambda_list[idx]

### it says optimal lamda is 1, let's try....

lambda <- 0.93

F <- matrix(0, nrow=p, ncol=p)
h <- matrix(0,nrow=p,ncol=1)

for (j in 0:(init-1)){
  F <- F + lambda^j * f(-j) %*% t(f(-j))
  h <- h + f(-j)*y_train[init-j]
}

# allocating space
theta.all <- matrix(NA,ncol=p, nrow=n)
sigma.all <- rep(NA, n)
sd.theta.all <- matrix(NA,ncol=p, nrow=n)
epsilon.all <- matrix(NA,nrow=n, ncol=1)


# solving at time init (0-100)
theta.hat <- solve(F)%*%h
theta.all[init,] <- theta.hat
epsilon.all[1:init] <- y_train[1:init] - t(f((init-1):0)) %*% theta.hat
sigma.all[init] <- sqrt(sum(epsilon.all[init]^2)/(init - p))
sd.theta.all[init,] <- sigma.all[init] * sqrt(diag(solve(F)))
```

```
y_est <- matrix(NA,nrow=n,ncol=1)


for (i in seq(from = init+1, to = n, by = l_step)){
  y_est[i] <- t(f(1)) %*% theta.hat
  F <- F + lambda^(i-1) * f(-i+1) %*% t(f(-i+1))
  h <- lambda* LInv %*% h + f(0)%*%y_train[i]

  theta.hat <- solve(F)%*%h
  theta.all[i,] <- theta.hat

  epsilon.all[i] <- y_train[i] - y_est[i]
  sigma.all[i] <- sqrt(sum(epsilon.all[i]^2)/(i - p))
  sd.theta.all[i,] <- sigma.all[i] * sqrt(diag(solve(F)))
}

par(mfrow=c(1,1), mar=c(3,3,1,0.2),mgp=c(2,0.7,0))

plot(y_train, type="l", col = "red", xlim = c(0,750), ylim = c(200,420))
lines(y_est,col="green")

y_test <- matrix(NA,nrow=n,ncol=1)
c <- 0

for (i in seq(from = -n+1, to = 0, by = l_step)){
  c <- c+1
  y_test[c] <- t(f(i))%*%theta.hat[]

}
```