LUNDS UNIVERSITET
Lunds Tekniska Högskola

PROJECT IN TIME SERIES ANALYSIS
FMSN45

Forecasting Normalized Difference Vegetation Index in Sudan

Name:	**Lucat-id:**
Ali Bakly	al7765ba-s
Davy Than	da0047th-s

2024 January

Contents

1 Introduction

In this paper, we will model and predict the precipitation and vegetation index (NVDI) in the Sudanese cities Geneina and Kassala. The precipitation data spans from 1960 to 1999 and is collected each month. Since the NVDI data is collected every 10 days, we firstly interpolate the precipitation to have 3 data points each month. The interpolated rain data will then be used as input for our Box-Jenkins model, where we predict the vegetation index in Geneina. We also compare the Box-Jenkins predictions with a SARMA model that does not use the precipitation as input, to see if the input data helped or not. To see if we can further improve the predictions we will recursively update our Box-Jenkins model so that our parameters can vary over time, using a Kalman filter. The model that best predicts the NVDI data in El Geneina will also be used in Kassala in order to see if our model works on other cities, and depending on the result why that is the case or not.

1.1 Data Selection

In figure 1 we see the normalized NVDI data for Geneina that we split to 75% modeling data, 15% validation data and 10% test data. We deem that the modeling data will be stationary enough after accounting for transformations. In the end of the modeling data we see a very strong spike which might mean that the dynamics of the data is changing, but we will resume with this data split.

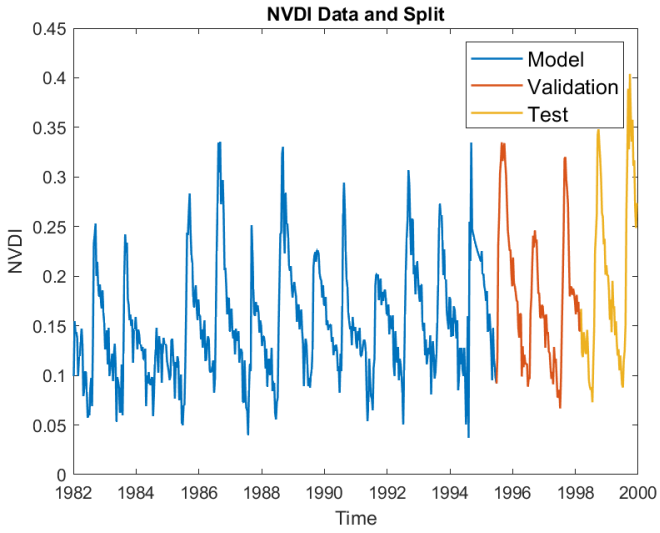


Figure 1: Normalized NVDI data from Geneina, split into model, validation, and test set.

1.2 Data Transformations

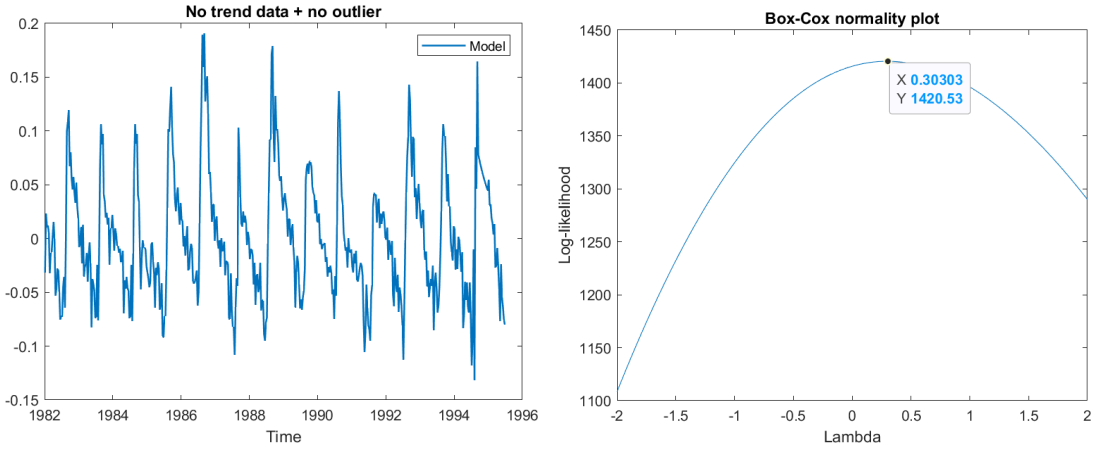
The NVDI data is given in the form of 8-bit data, which we transform to take values between in $[-1,\ 1]$:

$$2 \cdot \frac{y_t}{255} - 1.$$

In figure 1 one might notice a small trend. We handle this by removing the deterministic trend via a linear regression approach (MATLAB's fitlm). Of course, we only use the modeling set to get the intercept and slope, to avoid data leakage. We use the non trending data for all our modeling, and when it is time to predict, we add the trend back. Between 1983-1985 there was a drought in Sudan, causing the perceived outlier in the NVDI data around that time; there is no spike [1]. When using a

[1] Source: `https://pdf.usaid.gov/pdf_docs/pbaab327.pdf`

model with the rain as input, this should not be handled, since a low amount of rain would likely lead to a lower NVDI-index. However, when modeling the NVDI data without any input, this might affect the estimated model. In order to prevent this, we let 10 of the samples in that time period take the same values as 1 year before. This outlier handled data is only used in section 3.2. In figure 2 a) we find the non-trending modeling data with the outlier mentioned being handled. In figure 2 b) we also find the Box-Cox plot of the modeling data, which seems to suggest that some kind of transformation could be helpful. We tried a $\log(y_t + a)$ transformation, for some constant a that gave a Box-Cox plot which is maximized at 1. This did not improve any results, so we refrain from using a transformation.



(a) Modeling data with Removed trend and outlier.

(b) Box-Cox plot of a).

Figure 2: Data transformation.

1.3 Note on Implementation and Testing

If not explicitly mentioned we use MATLAB's pem for all estimation of parameters, which implements a prediction error minimization

$$\hat{\boldsymbol{\theta}}_{\text{PEM}} = \arg\min_{\boldsymbol{\theta}} \sum_{t} |\epsilon_{t+1|t}(\boldsymbol{\Theta})|^2.$$

Here, $\boldsymbol{\Theta}$ includes the model of the process and the samples available up until t and $\epsilon_{t+1|t}$ denotes the one-step prediction error.

It should be noted that all our Acf and Pacf plots start from lag 1, in order to better showcase the shortcomings of our models. If not explicitly mentioned we always use the Monti-test to test if the residuals are white and the D'Agostino's K-squared test to check if the Pacf values are normal distributed, since the Mont-test assumes that they are.

All the results on the test data were only computed after we had finished with our modeling using the model and validation sets.

2 Rain Data Reconstruction

2.1 State Space Model

We are given monthly data of accumulated rain during each period, whereas the vegetation index was measured three times each month (day 1–10, 11–20, and 21–end). In order to use the rain data as

exogenous input it would need to be on the same time scale as the NVDI data. Let y_t denote the accumulated monthly rain data, and x_t the 10 day accumulated rain, then we can write the following relation,

$$y_t = x_t + x_{t-1} + x_{t-2} + w_t, \quad t = 1, 4, 7, ... \tag{1}$$

In figure 3 we see the original data points and a rather useless linear interpolation, since it does not sum up correctly.

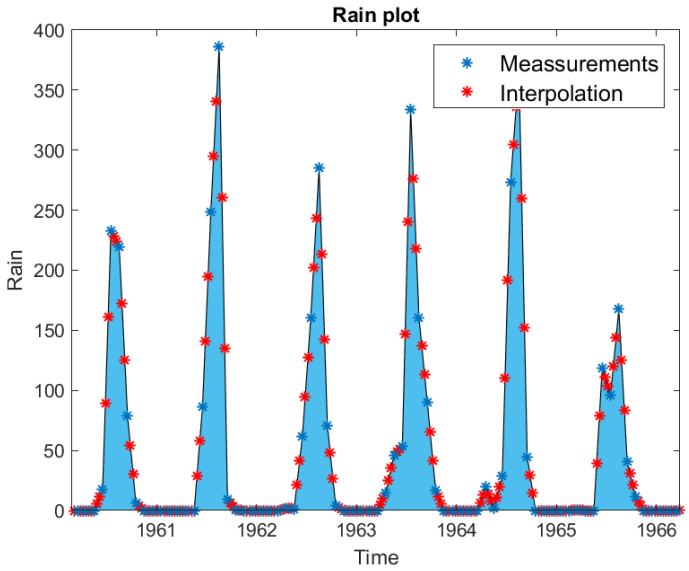


Figure 3: Excerpt of monthly rain data, linearly interpolated.

We have also been given additional information about the process x_t, namely that it should propagate as an AR(1) process. Our ultimate goal is to use a Kalman filter to interpolate our monthly rain data into the finer 10-day time scaled data. In order to do this we must set up a state-space model. There are more than a few ways to do this. One approach could be to handle the unknown values as missing values, and use the the known values divided by 3, with a state space model corresponding to the observable canonical form of an AR(1). If a sample is missing, we would just retain the earlier state. However, we did not adapt this approach. Instead we update the rain 3 times at a time by recursively applying the AR(1) formula:

$$\begin{aligned}
x_{t+1} &= -a_1 x_t + e_{t+1}, \\
x_{t+2} &= -a_1 x_{t+1} + e_{t+2} = a_1^2 x_t - a_1 e_{t+1} + e_{t+2}, \\
x_{t+3} &= -a_1 x_{t+2} + e_{t+3} = -a_1^3 x_t + a_1^2 e_{t+1} - a_1 e_{t+2} + e_{t+3}.
\end{aligned}$$

This, together with equation 1, yields the state space form

$$\underbrace{\begin{bmatrix} x_{t+3} \\ x_{t+2} \\ x_{t+1} \end{bmatrix}}_{\mathbf{x}_{t+1}} = \underbrace{\begin{bmatrix} -a_1^3 & 0 & 0 \\ a_1^2 & 0 & 0 \\ -a_1 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_t \\ x_{t-1} \\ x_{t-2} \end{bmatrix}}_{\mathbf{x}_t} + \underbrace{\begin{bmatrix} 1 & -a_1 & a_1^2 \\ 0 & 1 & -a_1 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{Z}} \underbrace{\begin{bmatrix} e_{t+3} \\ e_{t+2} \\ e_{t+1} \end{bmatrix}}_{\mathbf{e}_{t+1}}$$

$$y_t = \underbrace{\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}}_{\mathbf{C}} \underbrace{\begin{bmatrix} x_t \\ x_{t-1} \\ x_{t-2} \end{bmatrix}}_{\mathbf{x}_t} + w_t, \quad t = 1, 4, 7....$$

For the Kalman filter we need to provide a measurement noise covariance matrix $\mathbf{R}_w$ and a system noise covariance matrix, $\mathbf{R}_e$. Usually we get $\mathbf{R}_e$ from the simple definition

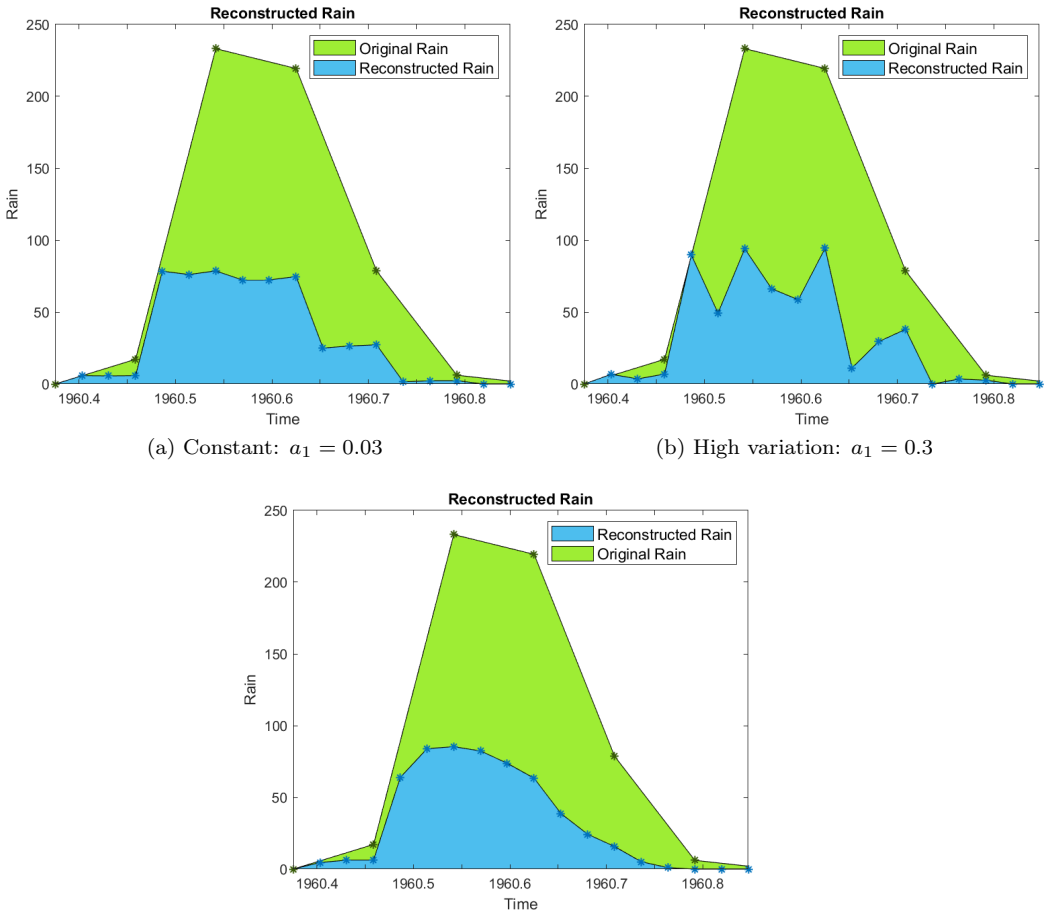
$$E\{\mathbf{e}_s\mathbf{e}_t^\mathsf{T}\} = \begin{cases} \mathbf{R}_e & \text{if } s = t, \\ \mathbf{0} & \text{otherwise,} \end{cases}$$

however the $\mathbf{Z}$ needs to be handled:

$$E\{\mathbf{Z}\mathbf{e}_s(\mathbf{Z}\mathbf{e}_t)^\mathsf{T}\} = E\{\mathbf{Z}\mathbf{e}_s\mathbf{e}_t^\mathsf{T}\mathbf{Z}^\mathsf{T}\} = \mathbf{Z}E\{\mathbf{e}_s\mathbf{e}_t^\mathsf{T}\}\mathbf{Z}^\mathsf{T} = \mathbf{Z}\mathbf{R}_e\mathbf{Z}^\mathsf{T}, \quad s = t. \tag{2}$$

2.2 Choosing the AR Parameter

The reconstructed rain on the finer time scale will obviously depend on what a_1 is chosen. One favorable situation is if a_1 can be chosen with the help of physical assumptions or prior knowledge. For example if one for some reason knows that the rain during the months are quite constant, very variable or change smoothly, you can choose a_1 accordingly. This is presented in figure 4.



(a) Constant: $a_1 = 0.03$

(b) High variation: $a_1 = 0.3$

(c) Smooth: $a_1 = -0.8$

Figure 4: Excerpt of rain reconstruction for different a_1 values.

We have no such prior knowledge and will not make such assumptions. Instead we will utilize the assumption that x_t should behave like an AR(1) process together with an EM(Expectation–maximization)-like approach. See Algorithm 1.

Algorithm 1 EM-like algorithm to estimate a_1

1: Initialize a_1 with an initial guess.
2: **while** a_1 not converged **do**
3: **Estimation:** Estimate $x(t)$ using Kalman filter and a_1
4: **Maximization:** Fit an AR(1) model to estimated $x(t)$ using PEM
5: **Update:** Update a_1 based on the fitted AR model
6: **end while**

The EM algorithm works by alternating between inferring the missing data (E-step) and optimizing the model parameters (M-step). In our approach, the Kalman filter serves the purpose of the E-step, estimating the hidden states (x_t) of the finer-scale rainfall data under the current parameterization of the AR(1) model. The subsequent M-step employs the Prediction Error Method (PEM) to update the AR(1) parameter (a_1) based on the estimated states. PEM, by minimizing the prediction error, indirectly optimizes the likelihood of the AR(1) model given the estimated states, which is conceptually aligned with the maximization of likelihood in traditional EM. Of course, we could set up the log likelihood function for the current estimate by looking at the residuals, and optimize for the best parameter, but we choose to use PEM for ease.

2.3 Notes on Implementation and Results

When determining the AR parameter through our EM approach, we only use the rain data on the same time span as the modeling set of the NVDI data, in order to prevent data leakage. Once we have convergence and the a_1 parameter is determined we perform the reconstruction on the whole time span of the original rain data. Our method quickly converges to $a_1 \approx -0.91$, somewhat independent of the initial a_1. If the initial parameter is chosen to $a_1 = 0.5$, our method converges in 9 iterations.

How quick this a_1 converges, and what the actual reconstruction looks like, will depend on the choices of $\mathbf{R}_e$ and $\mathbf{R}_w$. Assuming that the monthly rain data is accurate, we will choose $\mathbf{R}_w$ with small entries, $\mathbf{R}_w = \sigma_w^2\mathbf{I}$, where $\sigma_w^2 = 10^{-2}$. As we are not really sure how x_t behaves, and we do not trust the initial state values, we will choose $\mathbf{R}_e$ with larger entries, $\mathbf{R}_e = \sigma_e^2\mathbf{I}$, where $\sigma_e^2 = 10^2$. Of course we do not forget to account for $\mathbf{Z}$ in the system covariance matrix, as shown in equation 2. With these parameter we get results similar to figure 4 c), and for a few more years we show the result in figure 5.

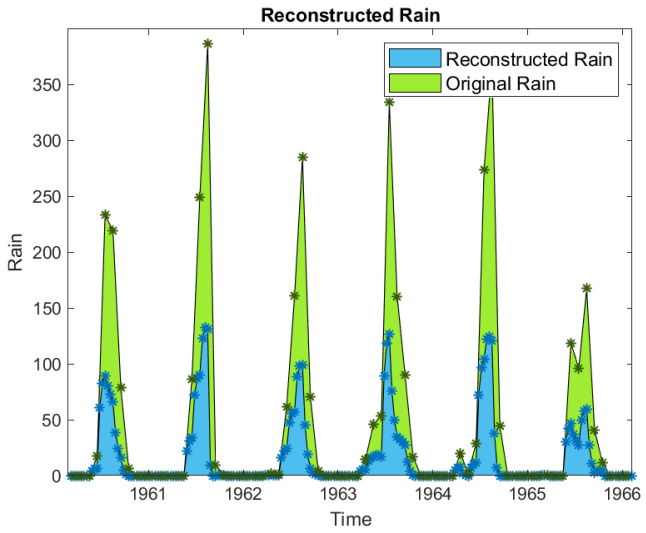


Figure 5: Excerpt of our reconstructed rain data.

From figure 5 it can also be seen that we do not have any negative values, which is favorable, at least if it should make sense in the physical world. There is no reason for the Kalman filter to not give negative values, instead we, ourselves, impose such a restriction. When a triplet of reconstructed 10 day data points include negative values, we want to find the closes triplet/vector with non negative values, that still has the same sum. This is essentially a simple case of NNLS (non-negative least squares),

$$\arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 \text{ subject to } \mathbf{x} \geq 0,$$

where in our case $\mathbf{A}$ would just be the identity matrix, with the additional sum constraint $\mathbf{1}^\mathsf{T}\mathbf{y} = \mathbf{1}^\mathsf{T}\mathbf{x}$. There are numerical methods for solving problems like this, but with $\mathbf{A} = \mathbf{I}$, the solution is trivial with

$$\mathbf{x} = \frac{\mathbf{1}^\mathsf{T}\mathbf{y}}{\mathbf{1}^\mathsf{T}\mathbf{z}}\mathbf{z}, \text{ where, } z_i = \max\{y_i, 0\} \ \forall i.$$

One way to make sure that the results are not totally wrong is to compare the sums, for example, the total sum of both the original data and the reconstructed data should be the same (or close). However this is quite a useless metric to infer if the reconstruction is actually any good. Take the interpolation which consists of the mean of the original data divided by 3, at every point on the finer time scale. The sums will be the same but the reconstruction will be useless. A better method is to take the vector $\mathbf{x}_{\text{rec}}$ which consists of the sum of all 10-day data triplets, this vector should be equal (or close) to the original rain data. We calculate the distance between the vectors as the 1-norm,

$$\|\mathbf{x}_{\text{org}} - \mathbf{x}_{\text{rec}}\|_1 \approx 0.0961,$$

quite a low number, which is good. In actuality, this number can be made arbitrarily small by decreasing σ_w^2 in $\mathbf{R}_w$, since the process in equation 1 would cease to be stochastic if $\sigma_w^2 = 0$.

3 Modeling of Nvdi With and Without Input

3.1 The Naive Predictor

To compare with a baseline, we create two naive predictors, one for the 1-step prediction and one for the 7-step prediction. The 1-step naive predictor simply predicts the current value as the next value. The 7-step predictor, however predicts the value from 1 year ago, due to the present seasonality. Their performances in terms of normalized variance of prediction residuals are given in table 1.

Table 1: Normalized prediction residual variances from Naive models.

	1-step	7-step
Validation	0.1463	0.7616
Test	0.1402	0.1673

3.2 SARMA Modeling Without Input

We will start by modeling the NVDI data without any precipitation data. As mentioned in the introduction, there was a drought in Sudan 1983-1985, and thus we have values that are outliers. To solve this we inserted the NVDI value of the year before the corrupted data points. It is by no means a perfect solution but since it does not affect many data points (and thus not super important in finding the model parameters) it is not worth spending resources on implementing a better solution.

3.2.1 Model Selection

Our final model ended up being of the form $S(z)A(z)y(t) = C(z)e(t)$ and below shows how we derived the polynomials.

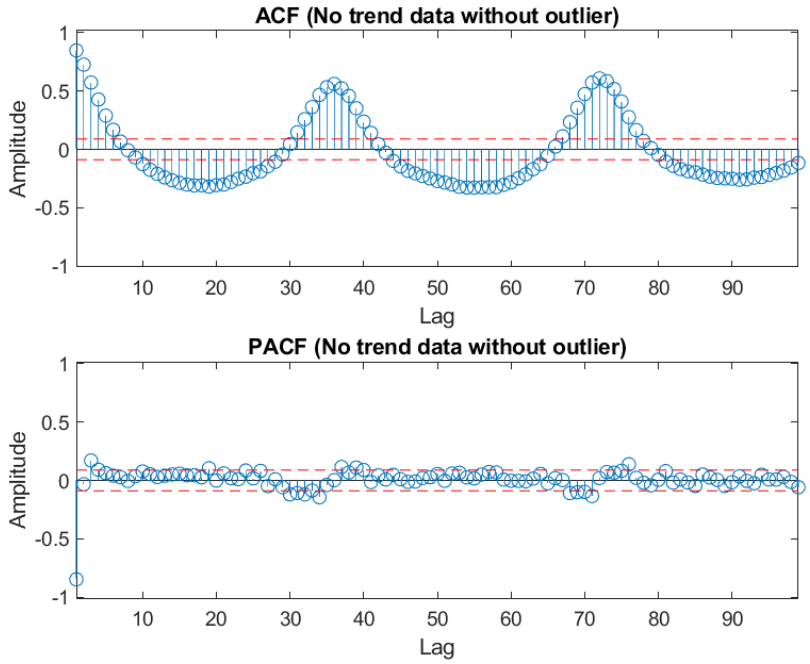


Figure 6: ACF and PACF plot on NVDI data

In figure 6 we see a strong seasonality of 36 in the ACF plot which is not surprising, since 36 lags represents a year in real time and it would be fair to assume that NVDI have a seasonality of a year. This is why we add a $S(z) = 1 + a_{36} \cdot z^{-36}$ term.

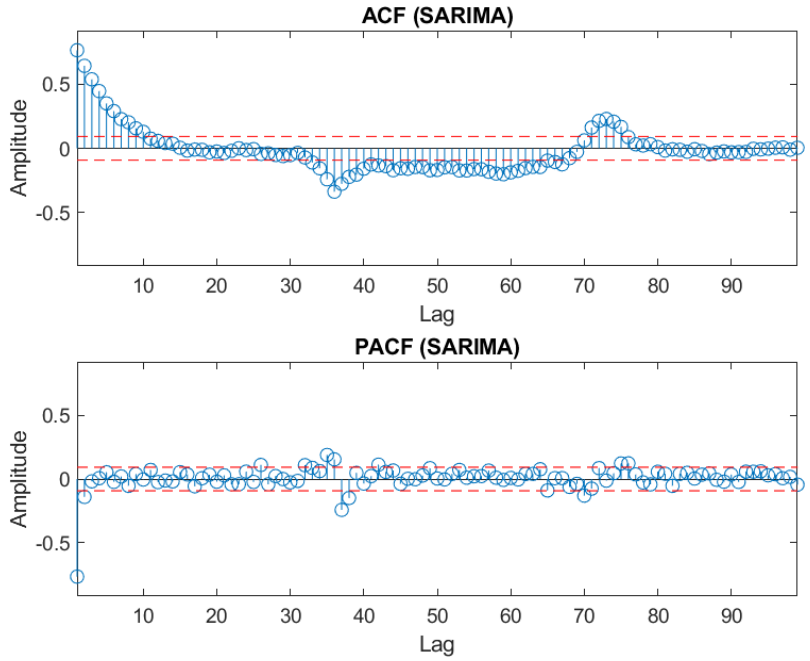


Figure 7: PACF and ACF of $S(z) = 1 - 0.6217(+/-0.03867)z^{-36}, A(z) = C(z) = 1$

Since the PACF at lag 1 in figure 7 shows a strong signal we add an AR(1) part $A(z) = 1 - a_1 z^{-1}$.

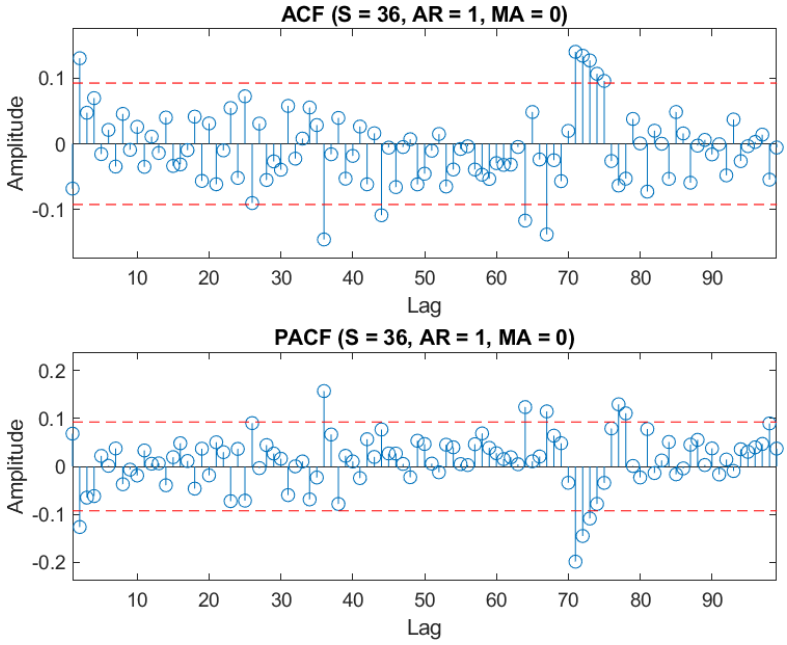


Figure 8: PACF and ACF of $S(z) \cdot A(z) = 1 - 0.7745(+/- 0.03013)z^{-1} - 0.3298(+/- 0.04912)z^{-36} + 0.185(+/- 0.05074)z^{-37}$

In figure 8 we see a strong peak at lag 36 and $\approx 36 \cdot 2$ in the ACF and thus we add a MA(36) part.

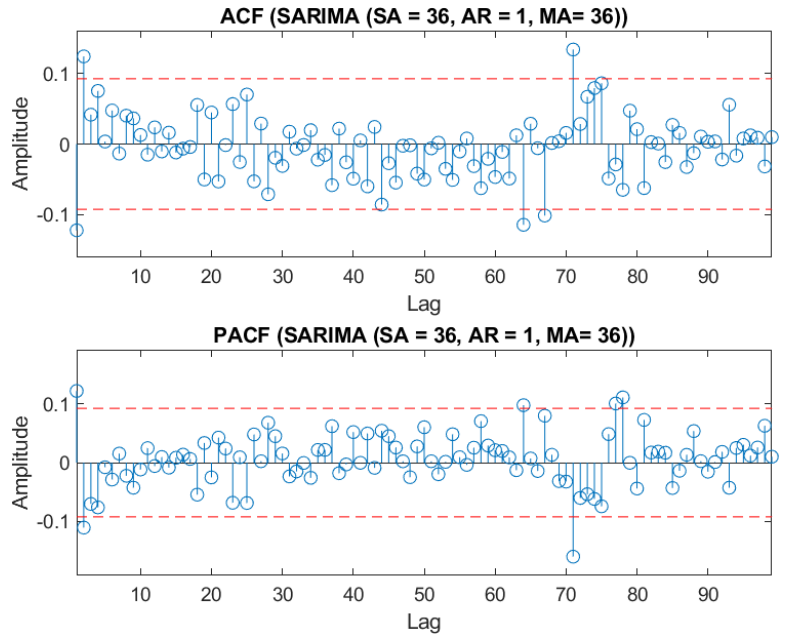
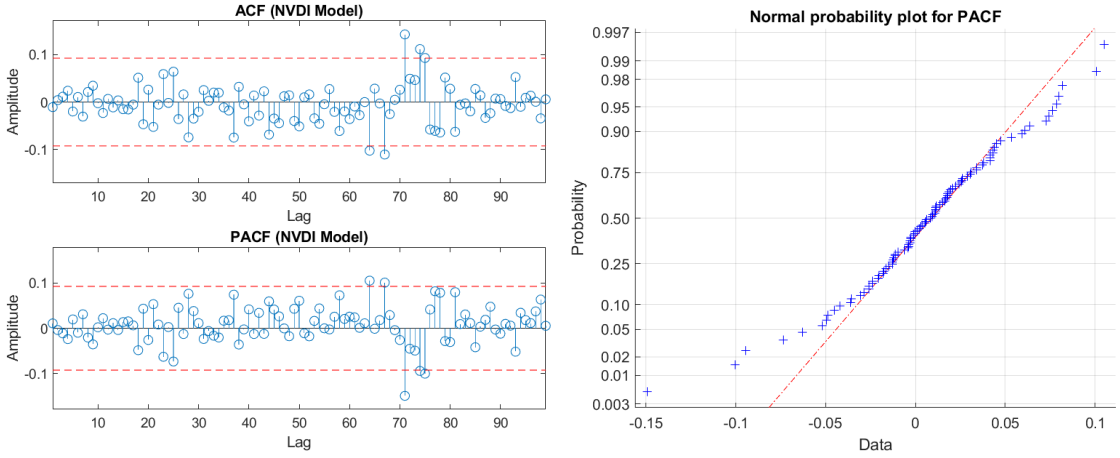


Figure 9: $S(z) \cdot A(z) = 1 - 0.697(+/- 0.03431)z^{-1} - 0.9804(+/- 0.04004)z^{-36} + 0.6608(+/- 0.05298)z^{-37}, C(z) = 1 - 0.8034(+/- 0.06309)z^{-36}$

We get a much better ACF in figure 9 than before but we still have a large out of the confidence interval peak at lags 1 and 2 and thus we expand our AR(1) part to an AR(2). This yields the final model seen

in figure 10, and model polynomials

$$\begin{aligned} S(z) \cdot A(z) &= 1 - 0.5781(+/-0.04799)z^{-1} - 0.1792(+/-0.04762)z^{-2} - 0.9822(+/-0.04205)z^{-36} \\ &\quad + 0.5146(+/-0.06746)z^{-37} + 0.2126(+/-0.05092)z^{-38}, \\ C(z) &= 1 - 0.8206(+/-0.05998)z^{-36} \end{aligned}$$



(a) Acf and Pacf.

(b) Normal probability plot of Pacf.

Figure 10: Our final SARMA model.

Our PACF errors are not normally distributed according to D'Agostino, but looking at figure 10 b) we can see that is not very far from being normal and because of this we can somewhat trust the Monti test a which says the error is white $4.64 < 31.41$. This means that we are done with our model.

3.2.2 Prediction

To test our model we have to predictions, which are done with equation 3. One thing to remember is that the deterministic trend we removed must now be added now added back to the data after the predictions are done. In figure 11 we have all of the plots for the predictions and in table 2 the performances are shown. The prediction is made as

$$\hat{y}_{t+k|t} = \frac{G(z)}{C(z)} y_t, \tag{3}$$

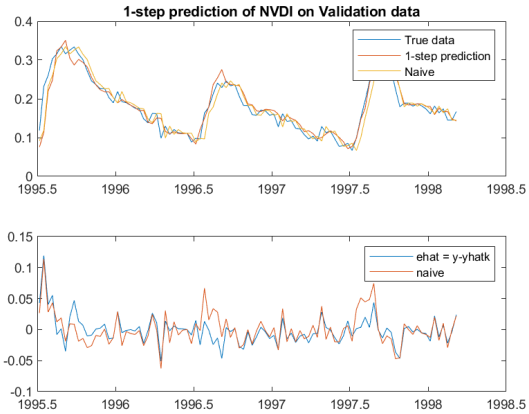
where $G(z)$ can be found from the polynomial division

$$\frac{C(z)}{S(z)A(z)} = F(z) + z^{-k}\frac{G(z)}{S(z)A(z)}.$$

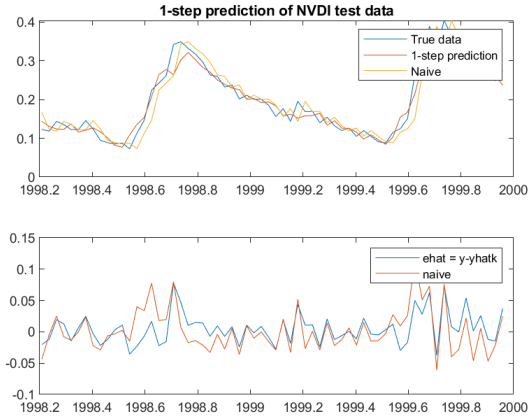
Of course, we do not forget to add back the removed trend, in order to plot in the correct domain.

Table 2: Resulting normalized prediction residual variances from SARMA model.

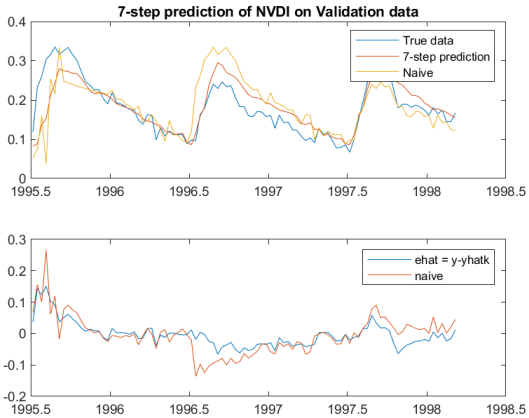
	1-step	7-step
Validation	0.1103	0.3395
Test	0.0756	0.1511



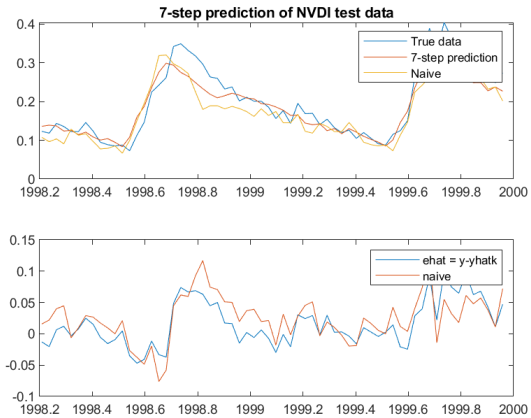
(a) 1-step prediction of NVDI validation data.



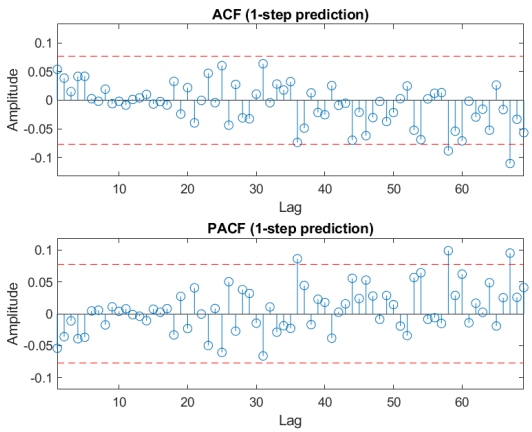
(b) 1-step prediction of NVDI test data.



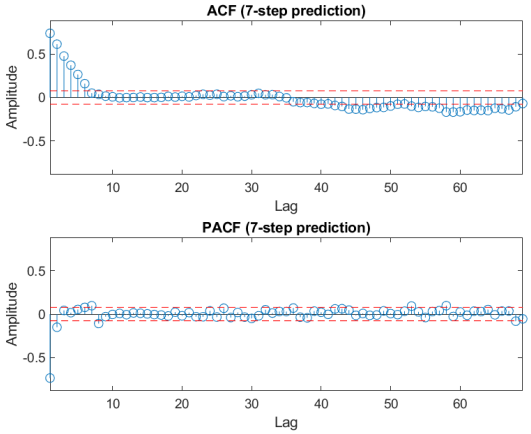
(c) 7-step prediction of NVDI validation data.



(d) 7-step prediction of NVDI test data.



(e) Acf and Pacf of 1-step prediction residuals.



(f) Acf and Pacf of 7-step prediction residuals.

Figure 11: Predicting the NVDI (without input) data.

We see that residuals of the 1-step predictor behaves like white noise, which Monti confirms, and that the 7-step predictor behaves like an MA(6). This coincides with the theory.

3.3 Box-Jenkins Model With Input

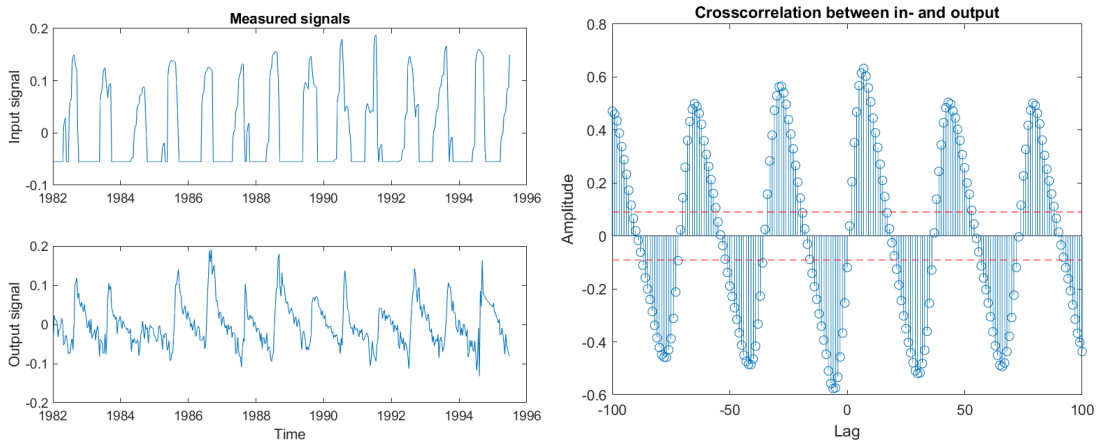
It might be reasonable to assume that introducing the reconstructed rain data as exogenous input yields an improved results. We will estimate a Box-Jenkins model,

$$y_t = \frac{B(z)}{A_2(z)} x_{t-d} + \frac{C_1(z)}{A_1(z)} e_t, \tag{4}$$

for this purpose. We found that transforming the rain data makes it more manageable to model and gives better performance. The transform is given by

$$\frac{\log(\mathbf{x}+1) - \bar{\mathbf{x}}}{20},$$

where $\bar{\mathbf{x}}$ denotes the mean of the input $\mathbf{x}$. The scaling, scales the input values to a similar range that the output values (NVDI) lies in. The transformed input values together with the NVDI data can be seen in figure 12 a). In figure 12 b) we see that there is actually correlation between the two signals; otherwise there would not be a reason to use model with exogenous input.



(a) The transformed input data compared to the transformed output data.

(b) Cross correlation between the rain data and the NVDI data.

Figure 12: Examining the data.

3.4 Modeling the Input

We now try to model the input data, by, at first, taking a look on the Acf and Pacf of the signal in figure 13. As seen in figure 13 there seems to be a strong AR(1) component together with a seasonality of 36. Upon testing, we found that a more complex model,

$$\begin{aligned} A_3(z) =& 1 - 1.046(+/-0.04696)z^{-1} + 0.3088(+/-0.04849)z^{-2} + 0.01657(+/-0.00798)z^{-12} \\ & - 0.9193(+/-0.03808)z^{-36} + 0.9616(+/-0.06414)z^{-37} - 0.3058(+/-0.05076)z^{-38}, \\ C_3(z) =& 1 - 0.07948(+/-0.04394)z^{-3} - 0.6224(+/-0.06235)z^{-36}, \end{aligned} \tag{5}$$

works better and gives favorable results in terms of white residuals. The Monti test shows that the residuals are white, which furthermore can be trusted since the Pacf passes as normal distributed according to D'Agostino's K-squared test. The Acf and Pacf of the residuals and the normplot of the Pacf are presented in figure 14.

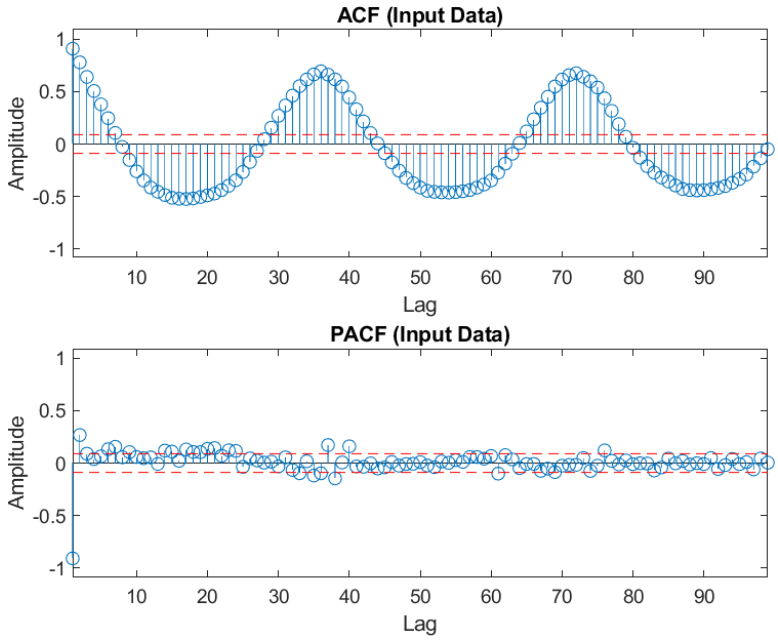
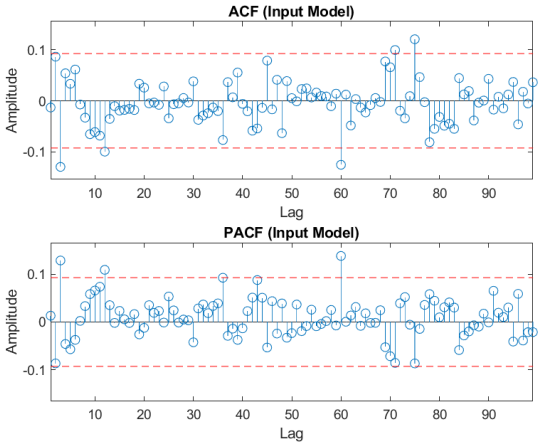
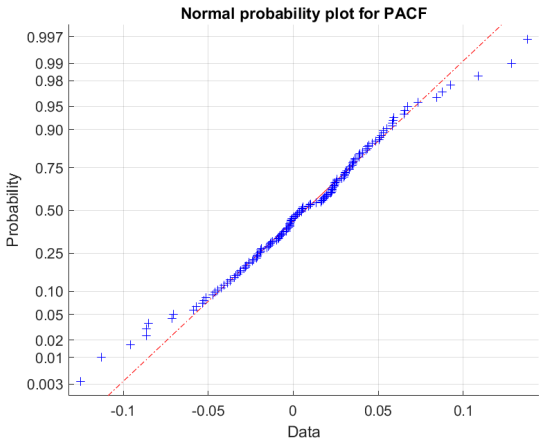


Figure 13: The Acf and Pacf of the transformed input.



(a) Acf and Pacf of the residuals from the model in equation 5.



(b) Normplot of the Pacf.

Figure 14: Modeling the input.

3.4.1 Model Selection

The model orders of the Box-Jenkins model can be found by examining the impulse response from the input signal to the output signal which can be estimated as the cross-correlation between ϵ_t and w_t, where

$$\epsilon_t = \frac{A_3(z)}{C_3(z)} y_t,$$
$$w_t = \frac{A_3(z)}{C_3(z)} x_t.$$

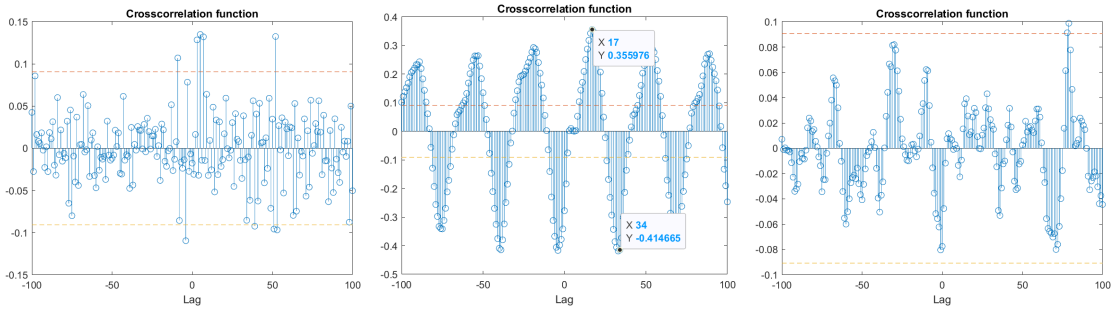
This procedure yields the cross correlation seen in figure 15 a). The plot does not resemble the usual cases for model order selection of the parameters d, r and s. For example we see that there is significant correlation at lags 3, 5 and 7, which would indicate $d = 3$, but it remains questioned what r and s would be in such a case. The method we go with instead, is to include significant parameters in $B(z)$ of the same order as the lag where we have significant correlation between the input x_t and $\tilde{e}_t$, where

$$\tilde{e}_t = y_t - \frac{B(z)z^{-d}}{A_2(z)}x_t.$$

The correlation between x_t and $\tilde{e}_t$ should be below the significance level, if one has succeeded in removing the influence of x_t on y_t. We start by including b_3z^{-3}, b_5z^{-5} and b_7z^{-7}, whose result is seen in 15 b), and iteratively add significant parameters, with the method we described. This led us to choose $B(z)$ of the form

$$B(z) = b_3z^{-3} + b_5z^{-5} + b_7z^{-7} + b_{17}z^{-17} + b_{31}z^{-31} + b_{34}z^{-34}. \tag{6}$$

We also found that $r = 0 \implies A_2(z) = 1$ gave the best results, in terms of the performance of the model. The cross-correlation between x_t and $\tilde{e}_t$ in figure 15 c), shows that we have successfully removed the influence of x_t on y_t.



(a) Cross-correlation between ϵ_t and w_t.

(b) Cross-correlation between x_t and $\tilde{e}_t$ showing strong peaks at lag 17 and 34 when only including, b_3z^{-3}, b_5z^{-5}, b_7z^{-7} and $r = 0$.

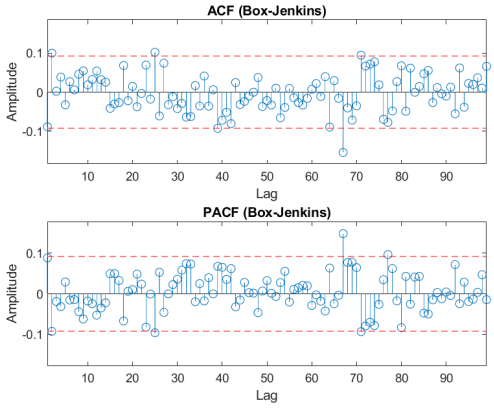
(c) Cross-correlation between x_t and $\tilde{e}_t$ after estimating the model in equation 6.

Figure 15: Modeling order selection.

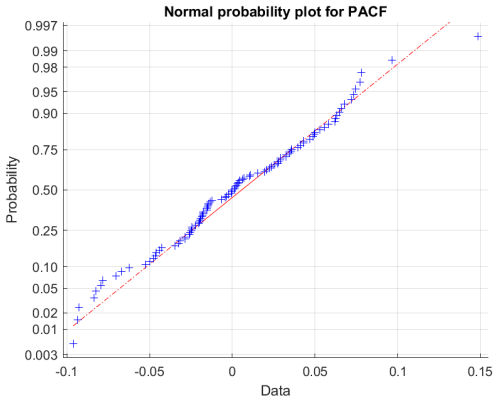
After this, we model $\tilde{e}$ as a very simple AR(1). The model yields white residuals and normally distributed Pacf values by the Monti test and the D'Agostino's K-squared test. We omit plots for this for conciseness. This means that the model order of $A_1(z)$ is 1 and that $C_1(z)$ is not included in the Box-Jenkins model, like $A_2(z)$. Having all the model orders, we can now fit a Box-Jenkins model, as described in equation 4, using MATLAB's pem.

$$\begin{aligned}
A_1(z) &= 1 - 0.6797(+/-0.03517)z^{-1}, \\
A_2(z) &= 1, \\
B(z) &= 0.1156(+/-0.03939)z^{-3} + 0.2157(+/-0.04168)z^{-5} + 0.1819(+/-0.03964)z^{-7} \\
&\quad + 0.07954(+/-0.03904)z^{-17} - 0.1217(+/-0.03845)z^{-31} - 0.1869(+/-0.03724)z^{-34} \\
C_1(z) &= 1.
\end{aligned} \tag{7}$$

This results in a model whose residuals pass as white and Pacf values pass as normal according to Monti and D'Agostino, which is further shown in figure 16. In figure 16 c) we also find that the residuals $\hat{e}_t$ from the final Box-Jenkins model is uncorrelated with the input signal x_t.



(a) Acf and Pacf of residuals from the estimated Box-Jenkins Model.



(b) Normplot of the Pacf values.

(c) Cross-correlation the input and the Box-Jenkins residuals.

Figure 16: Modeling order selection.

3.4.2 Prediction

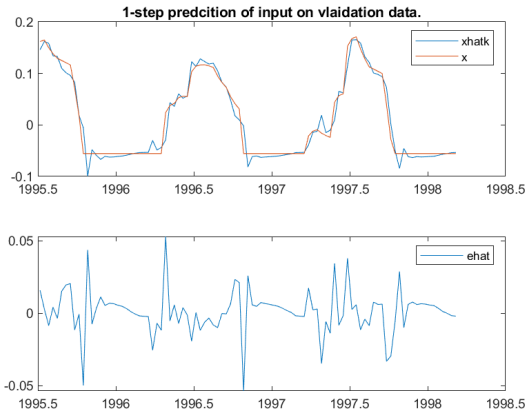
We can now use the found Box-Jenkins Model in equation 7 to make predictions on the validation and test data. Of course, to make a k-step prediction on the NVDI-data, we first need to make a k- step prediction on the rain data. The k-step prediction on the rain data is found as

$$\hat{x}_{t+k|t} = \frac{G_x(z)}{C_3(z)} x_t,$$

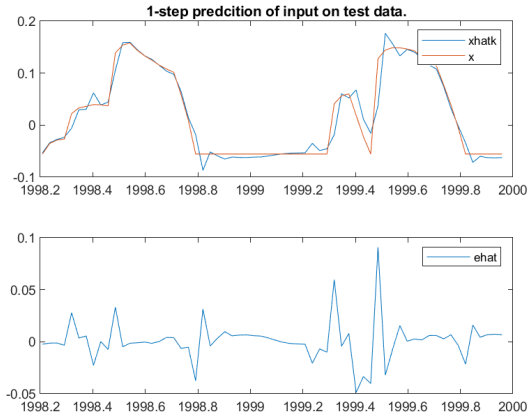
where $G_x(z)$ can be found from the polynomial division

$$\frac{C_3(z)}{A_3(z)} = F_x(z) + z^{-k}\frac{G_x(z)}{A_3(z)}.$$

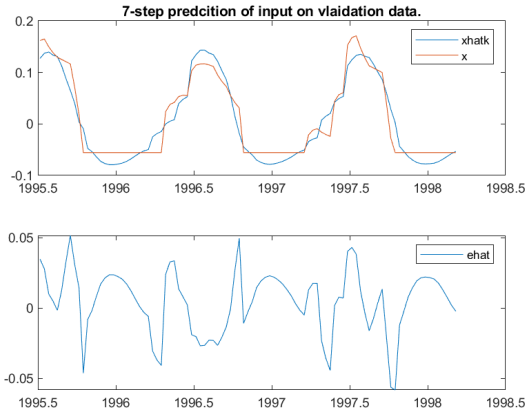
Doing this, yields the results presented in figure 17, where we see that the Acf and Pacf of the 1-step prediction residuals resemble white noise, and this is reassured by the Monti-test. The Acf of the 7-step prediction residual should indicate an MA(6), which it somewhat resembles, but not exactly.



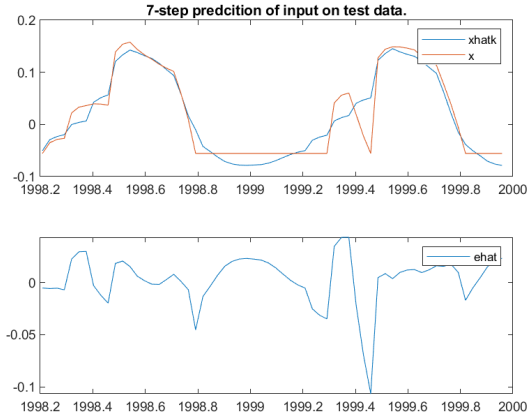
(a) 1-step prediction of input validation data.



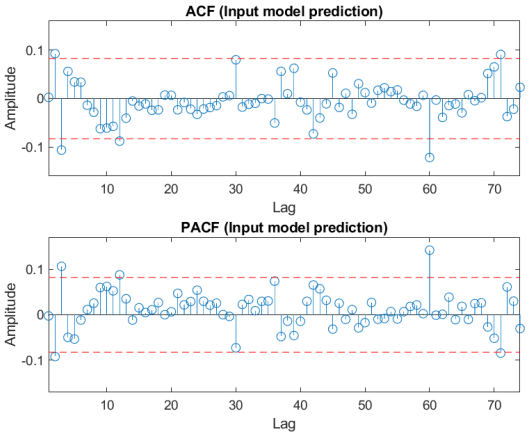
(b) 1-step prediction of input test data.



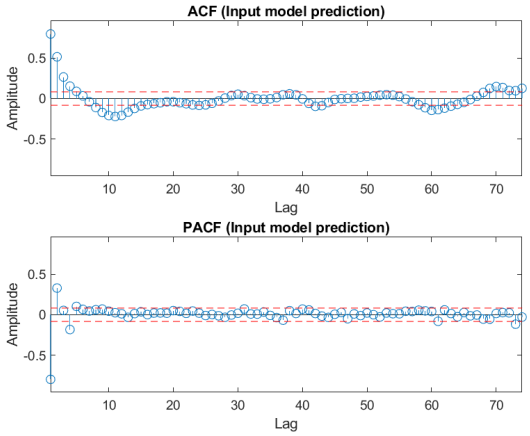
(c) 7-step prediction of input validation data.



(d) 7-step prediction of input test data.



(e) Acf and Pacf of 1-step prediction residuals.



(f) Acf and Pacf of 7-step prediction residuals.

Figure 17: Predicting the transformed input.

Now we can predict the NVDI data with

$$\hat{y}_{t+k|t} = \hat{\hat{F}}(z)\hat{x}_k + \frac{\hat{\hat{G}}(z)}{K_C(z)}x_t + \frac{G_y(z)}{K_C(z)}y_t,$$

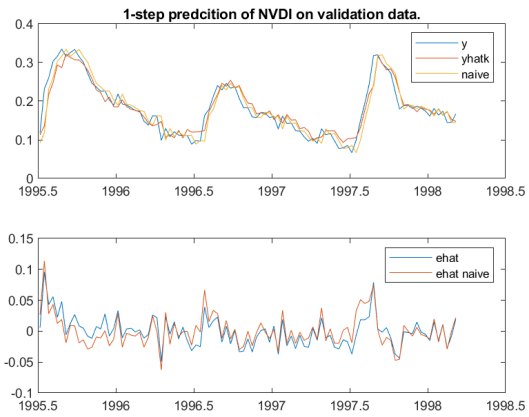
where

$$\frac{F_y(z)K_B(z)}{K_C(z)} = \hat{\hat{F}}(z) + z^{-k}\frac{\hat{\hat{G}}(z)}{K_C(z)},$$
$$\frac{C_1(z)}{A_1(z)} = F_y(z) + z^{-k}\frac{G_y(z)}{A_1(z)},$$

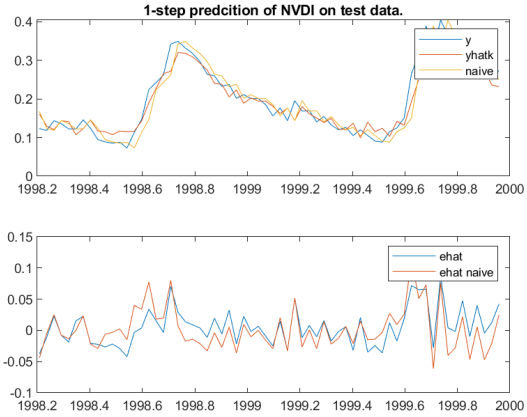
and

$$\begin{aligned} K_A(z) &= A_1(z)A_2(z), \\ K_B(z) &= A_1(z)B(z)z^{-d}, \\ K_C(z) &= A_2(z)C_1(z). \end{aligned} \tag{8}$$

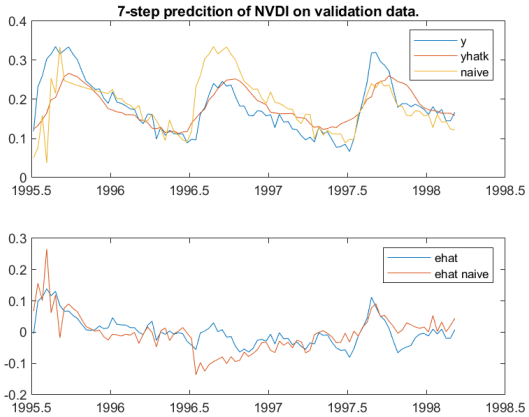
We, of course, do not forget to account for the removed trend, and plot the predictions, and the residuals in the original domain. The resulting predictions are seen in figure 18, where we, in e) see that the 1-step prediction looks white, and that the 7-step prediction looks like an MA(6) to begin with, but has some structure later.



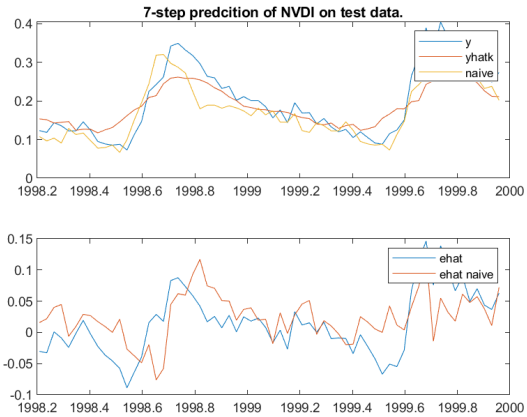
(a) 1-step prediction of NVDI validation data.



(b) 1-step prediction of NVDI test data.

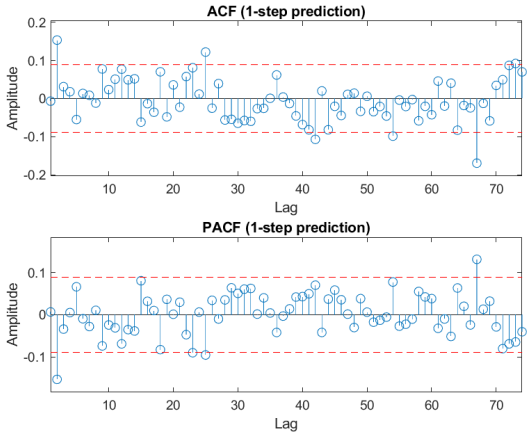


(c) 7-step prediction of NVDI validation data.

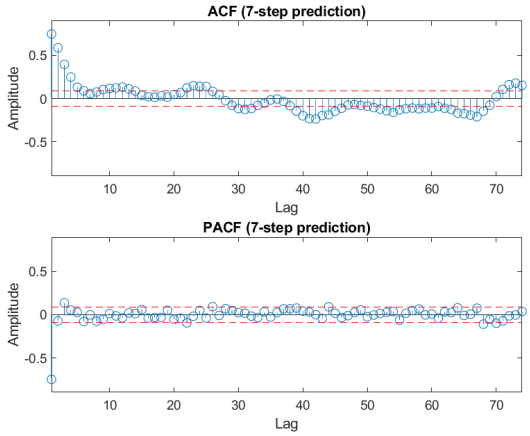


(d) 7-step prediction of NVDI test data.

Figure 18: Predicting the NVDI data using Box-Jenkins.



(e) Acf and Pacf of 1-step prediction residuals.



(f) Acf and Pacf of 7-step prediction residuals.

Figure 18: Predicting the NVDI data using Box-Jenkins.

We present the results in the form pf the variance of the prediction residuals normalized by the variance of the NVDI data on the same time span, in table 3. We find that the Box-Jenkins model (oddly) performs worse than the SARMA model, which we discuss in the last section.

Table 3: Resulting normalized prediction residual variances from Box-Jenkins model.

	1-step	7-step
Validation	0.1237	0.4345
Test	0.1042	0.3020

4 Dynamic Time Varying Parameter Estimation

It might be too strong of an assumption to assume that the NVDI data is stationary (even after all the transforms). In such a case, a model with time varying parameters might be more suitable and perform better. With the polynomials in equation 8, the Box-Jenkins model may be expressed as

$$K_A(z)y_t = K_C(z)e_t + K_B(z)x_t$$

which in turn can be written on state space form as

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{A}\mathbf{x}_t + \mathbf{e}_t, \\ y_t &= \mathbf{C}_{t|t-1}\mathbf{x}_{t|t-1} + w_t. \end{aligned}$$

The state vector $\mathbf{x}_t$ consists of the parameters to be estimated and the transition matrix $\mathbf{A}$ is nothing but the identity matrix $\mathbf{I}$. In order to make sense out of $\mathbf{C}$ we take a look at where K_A, K_B and K_C are non zero, and mark with x:

$$\begin{aligned} K_A(z) &= [1,\ x], \\ K_B(z) &= [0,\ 0,\ 0,\ x,\ x,\ x,\ x,\ ...], \\ K_C(z) &= [1]. \end{aligned} \tag{9}$$

This means that

$$\mathbf{C}_{t|t-1} = \begin{bmatrix} -y_{t-1} & x_{t-3} & x_{t-4} & x_{t-5} & x_{t-6} & x_{t-7} & ... \end{bmatrix}$$

We can now find the 1-step and 7-step predictions via

$$\hat{y}_{t+1|t} = \mathbf{C}_{t+1|t}\mathbf{A}\hat{\mathbf{x}}_{t|t} = \mathbf{C}_{t+1|t}\hat{\mathbf{x}}_{t|t},$$
$$\hat{y}_{t+7|t} = \mathbf{C}_{t+7|t}\mathbf{A}^7\hat{\mathbf{x}}_{t|t} = \mathbf{C}_{t+7|t}\hat{\mathbf{x}}_{t|t},$$

where

$$\mathbf{C}_{t+1|t} = \begin{bmatrix} -y_t & x_{t-2} & x_{t-3} & x_{t-4} & x_{t-5} & x_{t-6} & ... \end{bmatrix},$$
$$\mathbf{C}_{t+7|t} = \begin{bmatrix} -\hat{y}_{t+6|t} & \hat{x}_{t+4|t} & \hat{x}_{t+3|t} & \hat{x}_{t+2|t} & \hat{x}_{t+1|t} & x_t & ... \end{bmatrix}.$$

We see that the 1-step prediction $\mathbf{C}$-matrix is nice, in the sense that we do not need future values. For the 7-step prediction $\mathbf{C}$-matrix we need both future NVDI values and rain values. In practice this is done in a loop where we iterate,

$$\mathbf{C}_{t+1|t}, \hat{y}_{t+1|t} \implies \mathbf{C}_{t+2|t}, \hat{y}_{t+2|t} \implies ... \implies \mathbf{C}_{t+7|t}, \hat{y}_{t+7|t}.$$

However, we have the added difficulty of needing k-step predictions for the input, for $k = 1,\ 2,\ 3,\ 4$. This is handled prior by setting up a separate Kalman filter for the input where we make these predictions that later will be directly used in $\mathbf{C}_{t+k|t}$ for $k = 4,\ 5,\ 6,\ 7$. In order to set up the Kalman filter for the input we construct the controllable canonical form for the model of the input in equation 5:

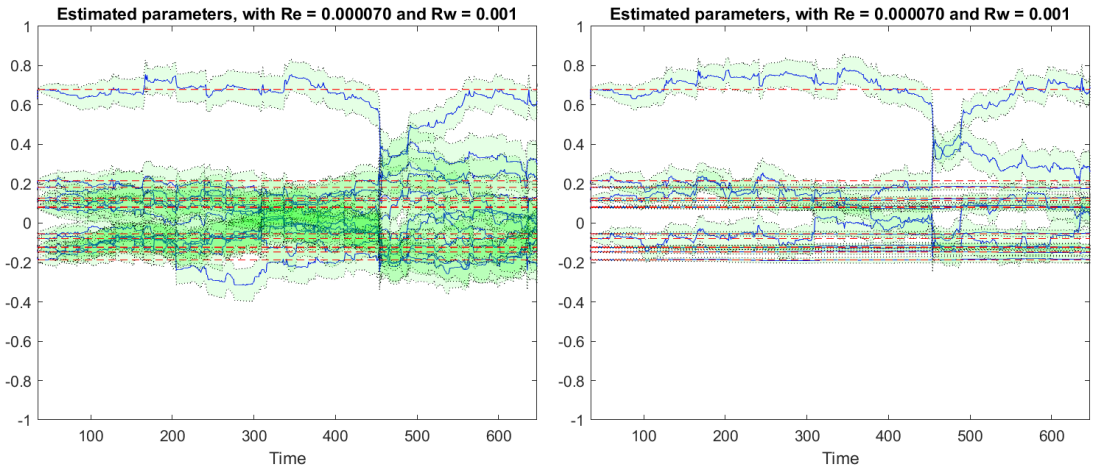
$$\tilde{\mathbf{x}}_t = \underbrace{\begin{bmatrix} -a_1 & -a_2 & \dots & -a_{d-1} & -a_d \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}}_{\tilde{\mathbf{A}}} \tilde{\mathbf{x}}_{t-1} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tilde{e}_t$$
$$x_t = \underbrace{\begin{bmatrix} 1 & c_1 & \dots & c_{d-1} \end{bmatrix}}_{\tilde{\mathbf{C}}} \tilde{\mathbf{x}}_t$$

where $d = \max(p, q+1)$, and $a_l = 0$, for $l > p$, and $c_l = 0$ for $l > q$. For the input, the k-step prediction is found as

$$\hat{x}_{t+k|t} = \tilde{\mathbf{C}}\tilde{\mathbf{A}}^k\hat{\tilde{\mathbf{x}}}_{t|t}.$$

Note that $\tilde{\mathbf{C}}$ is not time varying and that $\tilde{\mathbf{A}} \neq \mathbf{I}$. Also note that we use the stationary model for the input to recursively update the input predictions. Of course, one might prefer to also let the parameters of the input model be time varying.

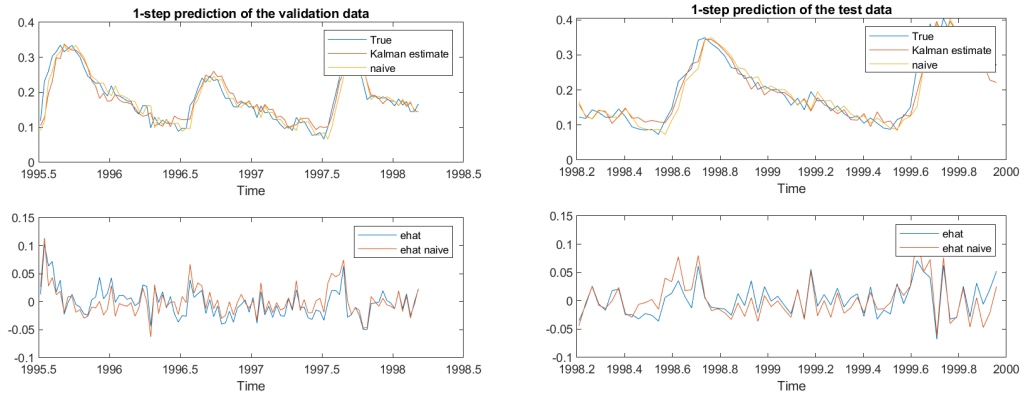
Now it is left to choose the covariance matrices and initial states. For the input we take $\tilde{\mathbf{R}}_e = \tilde{\mathbf{R}}^{x,x}_{1|0} = 10$ and $\tilde{\mathbf{R}}_w = 1$. All the initial states are set to 0. For predicting the NVDI-data we initially use $\mathbf{R}_e = \mathbf{R}^{x,x}_{1|0} = 7 \cdot 10^{-5}$ and $\mathbf{R}_w = 5 \cdot 10^{-4}$ for all parameters. The initial states are set to the parameter values estimated from the stationary model. The covariances let's the parameters vary quite a bit, which gave us bad performances. Instead we choose to give higher variances to the parameters who we deemed vary the most, while letting all other parameters change very little. We changed $\mathbf{R}_e$ to $\mathbf{R}_e = 7 \cdot 10^{-7}$ for the parameters we deemed not time varying. See the comparison in figure 19. Note the big jump around sample 450, this is due to the big spike in the end of the modeling NVDI data, which we mentioned in section 1.1!



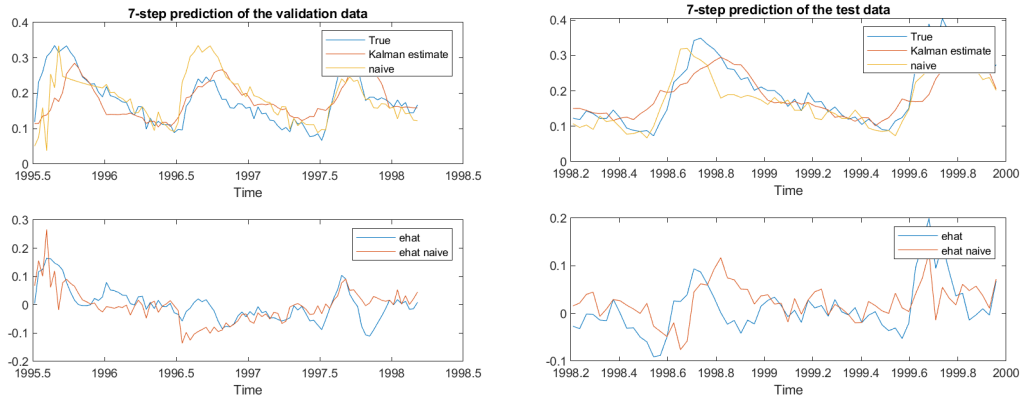
(a) Parameters when letting all be time varying. (b) Parameters when letting a few be time varying.

Figure 19: Parameter estimation with Kalman filter.

We present the resulting predictions in figure 20.

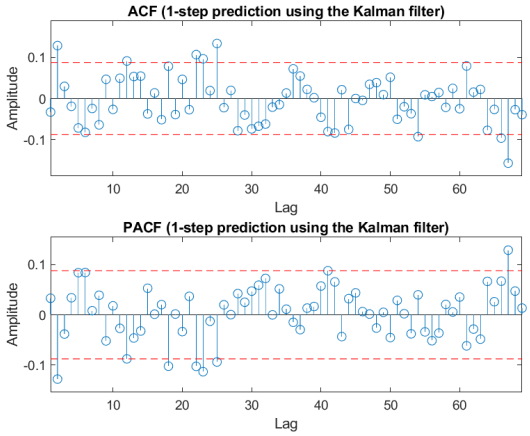


(a) 1-step prediction of NVDI validation data. (b) 1-step prediction of NVDI test data.

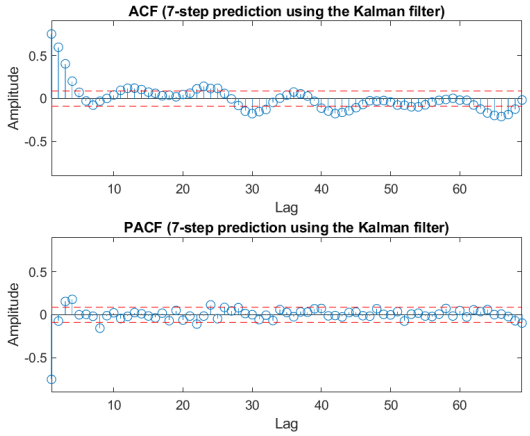


(c) 7-step prediction of NVDI validation data. (d) 7-step prediction of NVDI test data.

Figure 20: Predicting the NVDI data with dynamic time varying parameters using Kalman filter.



(e) Acf and Pacf of 1-step prediction residuals.



(f) Acf and Pacf of 7-step prediction residuals.

Figure 20: Predicting the NVDI data with dynamic time varying parameters using Kalman filter.

We find that the 1-step prediction resiudals are almost white with Monti scoring it $33.18 > 31.41$, while D'Agostino tells us that the Pacf is normally distributed. The 7-step prediction looks more like an MA(5) than an MA(6), and there are some structure after that, but nothing really out of the ordinary when it comes to finite data. We present the normalized residual variances in table 4.

Table 4: Resulting normalized prediction residual variances from Kalman filter with a few time varying parameters.

	1-step	7-step
Validation	0.1479	0.6926
Test	0.0998	0.3333

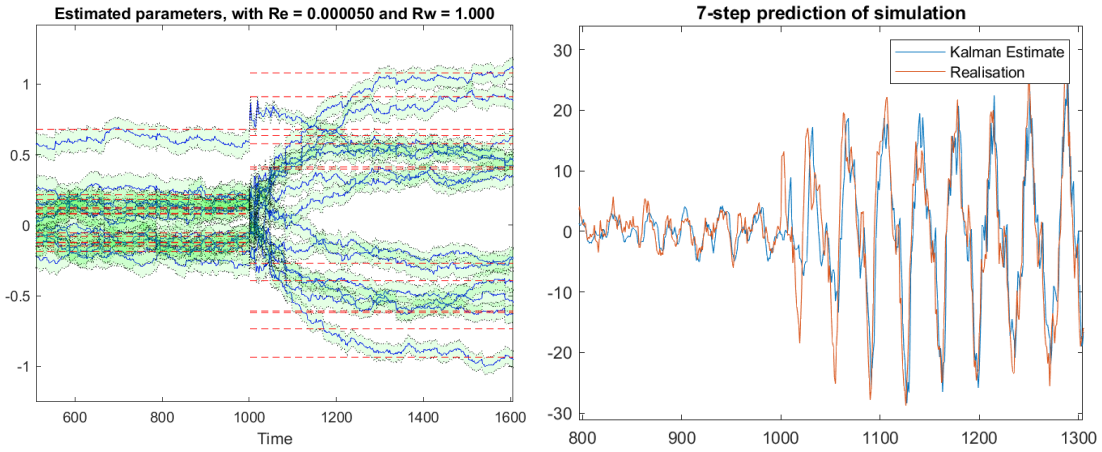
Comparing table 4 to table 3 we see that most of the results are similar but that the Kalman filter performs a lot worse on the validation data, when it comes to the 7-step prediction. We found that this is because the big jump around sample 450, around the start of the validation data. It seems that it is over correcting for this change. After testing many combinations on the Covariance matrices we found that it actually performs best when all the parameters are stationary which gave the results in table 5.

Table 5: Resulting normalized prediction residual variances from Kalman filter with stationary parameters.

	1-step	7-step
Validation	0.1237	0.5259
Test	0.1039	0.3150

As expected, these values are on par with the stationary Box-Jenkins model. We must admit that we find it odd that the Kalman filter does not perform better. It could just be that we have found good parameters that describe the data well enough even when not letting them vary over time. A fair question would be if we might have a bug in our implementation, and to that question we would be inclined to answer no. After simulating a realization with the found model parameters, and adding a step in some of those, we executed the same code for the Kalman filter. As seen in figure 21 a) the filter does seem to adapt to the new parameters. In figure 21 b) we see that the 7-step prediction does seem

to work.



(a) Estimated parameters for a simulated realization adapting to the step.

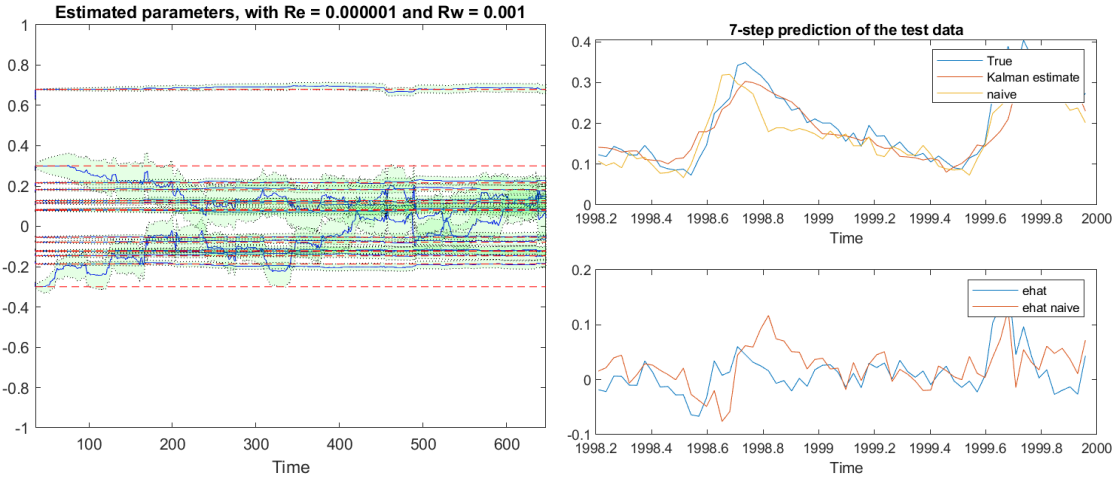
(b) The 7-step prediction when the dynamics of the data changes.

Figure 21: Parameter estimation and 7-step prediction with Kalman filter of simulated data.

The last thing that might affect the performance is that we need more complexity in the $K_A(z)$ and $K_C(z)$ polynomials. As seen in 9 these are very simple, and their might be more parameters that vary over time, but that are not apparent in stationary modeling A reasonable assumption is that the predictions also depend on the value and noise from 1 year prior(36 samples), accounting for this, we instead get

$$\mathbf{C}_{t|t-1} = \begin{bmatrix} -y_{t-1} & -y_{t-36} & x_{t-3} & x_{t-4} & x_{t-5} & x_{t-6} & x_{t-7} & \dots & e_{t-36}. \end{bmatrix}$$

Using this model instead, and only letting the newly introduced parameters vary, yields improved results seen in figure 22 and in table 6.



(a) Estimated time varying parameters with modified model.

(b) The 7-step prediction on the test data with modified model.

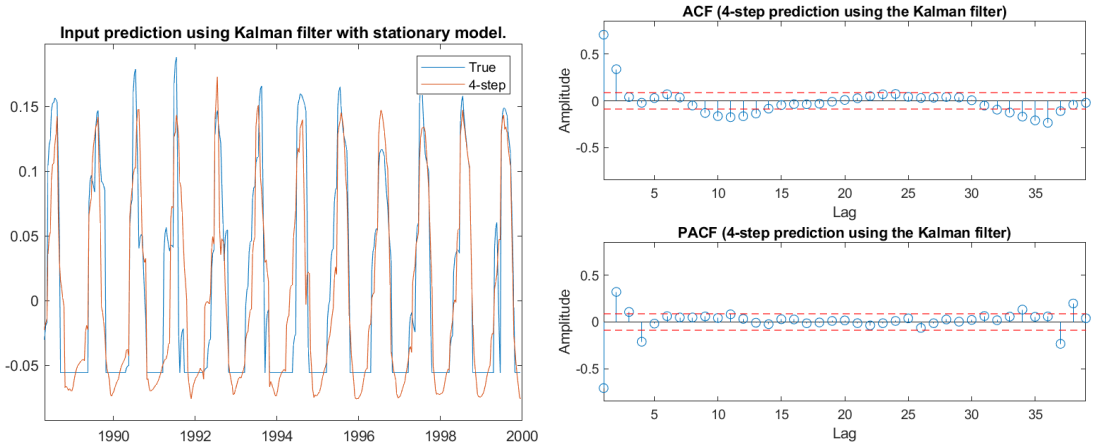
Figure 22: Parameter estimation and 7-step prediction using Kalman filter with modified model.

Table 6: Resulting normalized prediction residual variances from Kalman filter with modified model

	1-step	7-step
Validation	0.1354	0.6543
Test	0.0891	0.1912

We find for the test data that the 1-step prediction is improved with $\approx 15\%$ and that the 7-step prediction is improved with $\approx 37\%$, as compared to the stationary model in table 3. In figure 22 a) the estimated parameters can be seen varying and in b) the 7-step prediction on test data is presented. As for the validation data, and the Acf's and Pacf's, they are similar to the ones presented in 20, so we omit these plots.

One last note that we should make, is that we did actually also implement a Kalman filter for the input with time varying parameters, as opposed to the the controllable canonical form we mentioned earlier. The performance was on par with the stationary model, so we omit this from the project. The 4-step prediction (which we at most need) of the input which is made by the Kalman filter, using the described controllable canonical form is seen in figure 23. The input prediction looks sufficient enough.The acf which in theory should indicate an MA(3) looks more like an MA(2), but could pass as an MA(3).



(a) The 4-step prediction of the input using the Kalman filter.

(b) The Acf and Pacf of the input 4-step prediction using the Kalman filter.

Figure 23: Input prediction with Kalman filter

5 Model Performance on the City of Kassala

Our best performing model was the stationary SARMA model, which we discuss in the next section. Therefore, we investigate this models performance on the NVDI data of the city of Kassala in Sudan. We present the results in table 8 and 9. We also provide how the naive predictor predictor performs on the Kassala data in table 7.

Comparing the SARMA model results (table 8 and table 9) to the SARMA model for El Geneina we see that the model has performed worse for Kassala although with re-estimated parameters the model worked a bit better. Basically, the model structure does not work as well for Kassala and there can be numerous reasons for it. First and foremost the NVDI data for Kassala differs from the El Geneina data, which can be caused by the different locations of the cities. Kassala is located much more near

the coast which can lead to more humidity which might or might not influence NVDI.

Table 7: Resulting normalized prediction residual variances from Naive model.

	1-step	7-step
Validation	0.2593	1.3600
Test	0.4129	0.9623

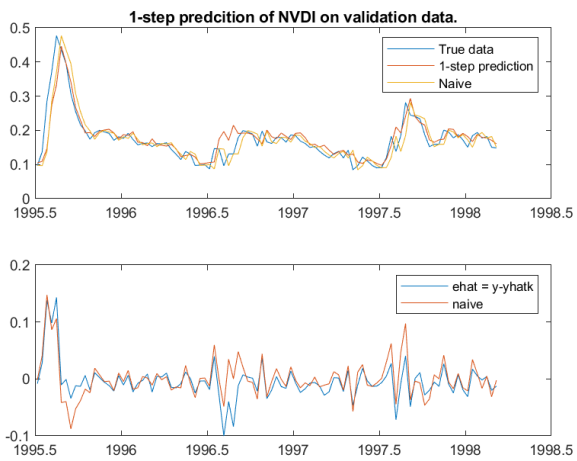
Table 8: Resulting normalized prediction residual variances from non-reestimated SARMA model.

	1-step	7-step
Validation	0.2561	0.7537
Test	0.3165	0.5072

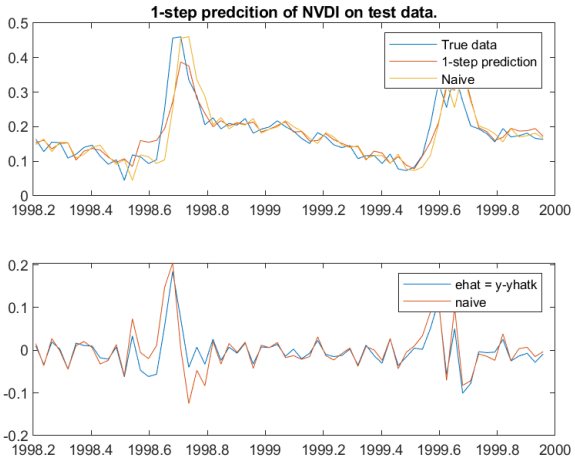
Table 9: Resulting normalized prediction residual variances from re-estimated SARMA model.

	1-step	7-step
Validation	0.2309	0.7025
Test	0.2750	0.4651

In, addition to the SARMA model, we did test our Box-Jenkins and recursive Box-Jenkins models, which one would expect to perform better. However, they performed on par with the SARMA model, likely due to the model orders being incorrect. Finally we provide some plots of the predictions on the Kassala data, when we use the same SARMA model structure, but with re-estimated parameters, in figure 24

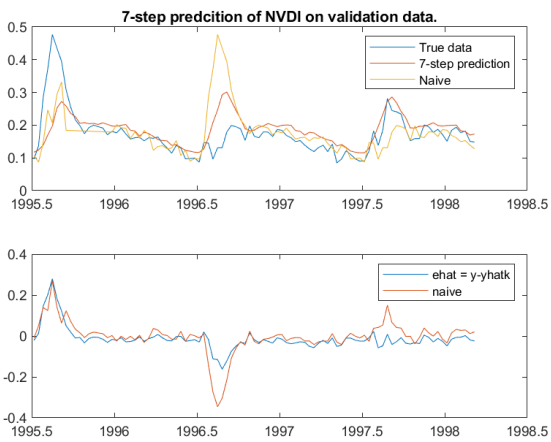


(a) 1-step prediction of NVDI validation data.

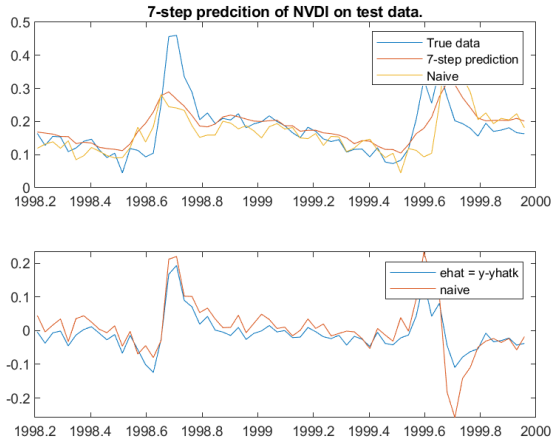


(b) 1-step prediction of NVDI test data.

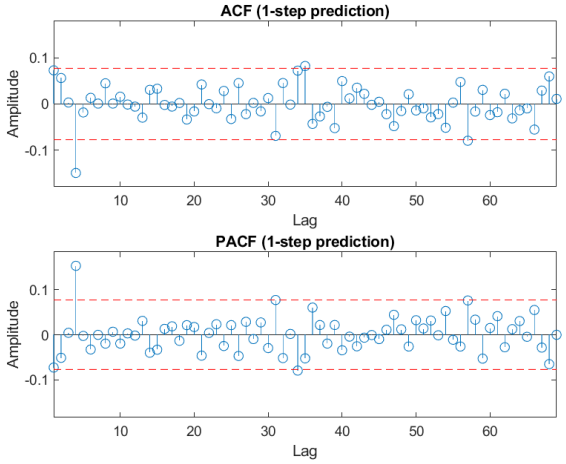
Figure 24: Predicting the NVDI data with dynamic time varying parameters using Kalman filter.



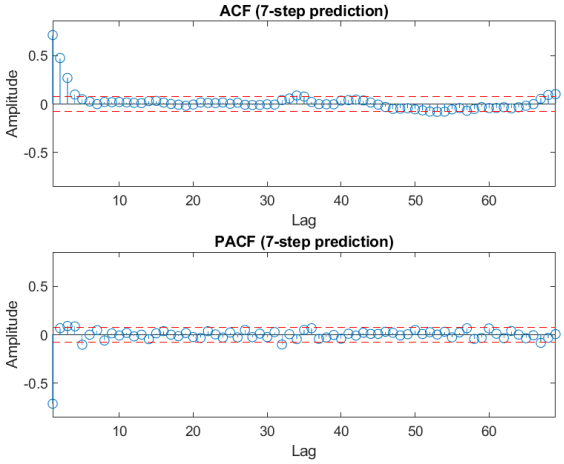
(c) 7-step prediction of NVDI validation data.



(d) 7-step prediction of NVDI test data.



(e) Acf and Pacf of 1-step prediction residuals.



(f) Acf and Pacf of 7-step prediction residuals.

Figure 24: Predicting the NVDI data of Kassala with re-estimated SARMA model.

We see that the 1-step prediction looks somewhat white, which Monti also confirms, but it should not be trusted since D'Agostino says that the Pacf values are not normally distributed. There is a strong component at lag 4. The 7-step prediction looks like an MA(3) or MA(4), in contrary to the theoretical MA(6), likely due to wrong model orders.

6 Discussion and Conclusion

We summarize our results from the different models on the validation data in table 10 and on the test data in table 11. We find that our stationary SARMA model does best on both the 1-step prediction and 7-step prediction, on the validation data. The Box Jenkins model does best on the validation data after the SARMA model, while the naive model does worst. We must admit that we find it odd that the Box-Jenkins model with input does not outperform our SARMA model. It might be that the NVDI, itself has clear structure which then can be modeled good enough, without any input. On the validation data, the dynamic Box-Jenkins(modified see table 6) does worse than the stationary Box-Jenkins. We explained this as a consequence of the dynamics changing rapidly right before the validation data starts which the Kalman filter picks up on. However, when the validation data starts the Kalman filter might

have overreacted previously, and therefore the performance on the validation suffers. As we see in the test data, the dynamic Box-Jenkins does much better.

Table 10: Resulting normalized prediction residual variances from the different models on the Validation data.

	1-step	7-step
Naive	0.1463	0.7616
SARMA	0.1103	0.3395
Box-Jenkins	0.1237	0.4345
Dynamic Box-Jenkins	0.1354	0.6543

Table 11: Resulting normalized prediction residual variances from the different models on the test data.

	1-step	7-step
Naive	0.1402	0.1673
SARMA	0.0756	0.1511
Box-Jenkins	0.1042	0.3020
Dynamic Box-Jenkins	0.0891	0.1912

On the test data we see that the SARMA model is still the best for the 1-step prediction, and beats the naive model with large margin. The dynamic Box-Jenkins is a close second for the 1-step prediction. For the 7-step data, however, the naive predictor is a strong competitor. SARMA is still the best for 7-step predictions, but the Naive model is not far off which beats both other models. In general we also note that the results on the test data is better than on the validation data, which might be surprising at first. However, this is not unexpected if one takes a look at figure 1. There we can see that the test data is very "nice" in the sense that it is well described by the value a year earlier, which is what the naive 7-step predictor does. Because of this we would not use the test set to infer that the Naive predictor is better than the two Box-Jenkins models.

We conclude that the stationary SARMA model is the best model overall, and does fairly well and beats our naive predictors with good margin. It would be interesting to further explore if the SARMA model would do better if it had time-varying parameters, similar to what we did for the Box-Jenkins model.