Project in Time Series Analysis

Carl Bernhardtz, Kalle Josefsson

December 2023

"All models are wrong but ours is useful."

- K.J 2023 Matteannexet

Contents

1 Introduction

2 Recursive reconstruction of rain data

The first task was to reconstruct the measurements of rain to get the measurements on the same time-scale as the NVDI data. We were instructed to model the rain as an AR(1) process.

2.1 Modeling rain as an AR(1)-process

Modeling the original rain data in El-Geneina as an AR(1)-process resulted in having $a_1 = -0.6119$.

2.2 Modeling rain as a different process

Using the aforementioned model resulted in what we considered to be a bad model of the rain. The reconstruction of the rain data later on using this model problems which we will discuss later. We therefore decided to try to create a better model for the rain which is describes below.

2.2.1 Data transformation

Firstly we examined the original rain data in order to assess if any necessary data transformation would be required. By plotting the original rain data we came to the conclusion that a power transformation could be beneficial to make the data more stationary. The original rain data is presented in figure 1.

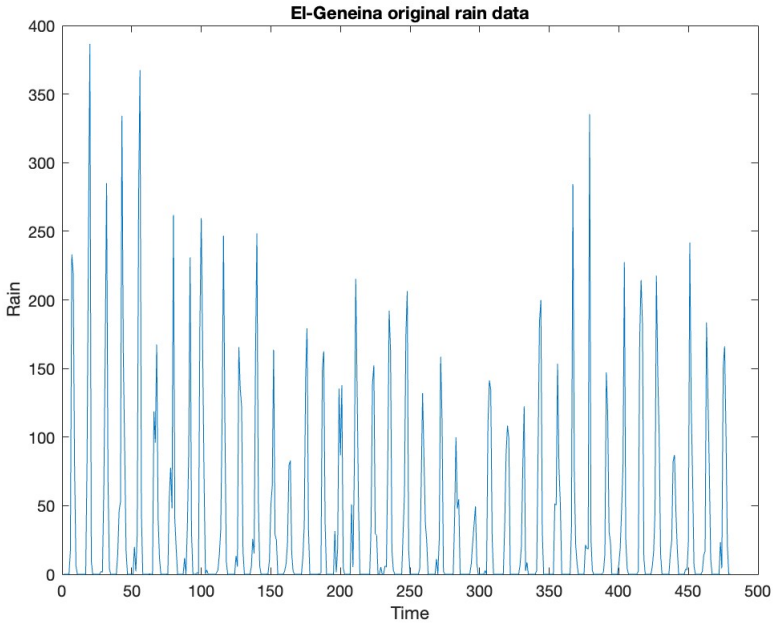


Figure 1: Data before transformation

To determine which transformation would be best we computed the maximum value of lambda using the Box-Cox normality plot which is presented in figure 2. This resulted in a $\lambda = -0.2626$. Comparing this to table 4.6 in the course book it is apparent that our lambda is between two of the standard transformations: $\lambda = -0.5$

and $\lambda = 0.0$. According to theory it is often sufficient to use the transformation of which the corresponding λ is closest to. Since our λ was more or less right in between we made the decision to use the log-transformation. Furthermore the data set contained several data points which are equal to zero which would need to be handled. By adding 1 to every data point we got an offset which evaded this problem.

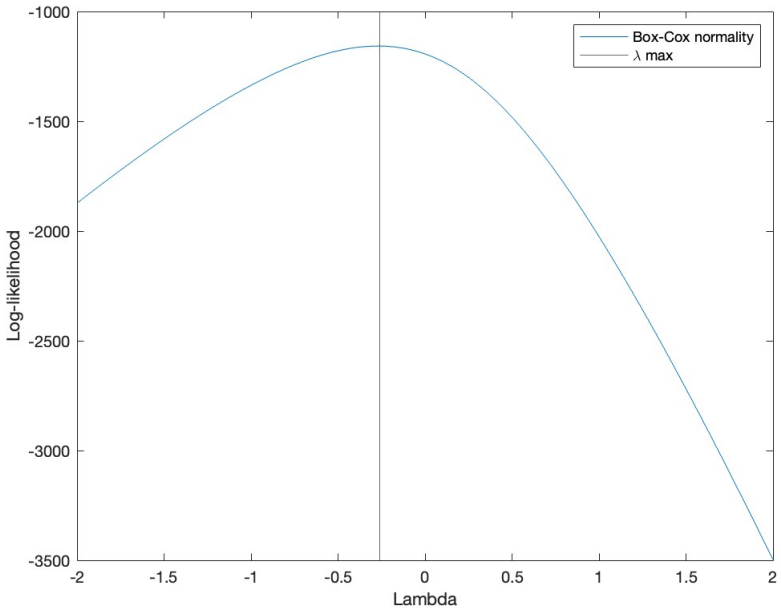


Figure 2: Box-Cox normality plot together with λ-max

The transformation yielded more stationary data which was what we wanted. The new transformed data is presented in figure 3.

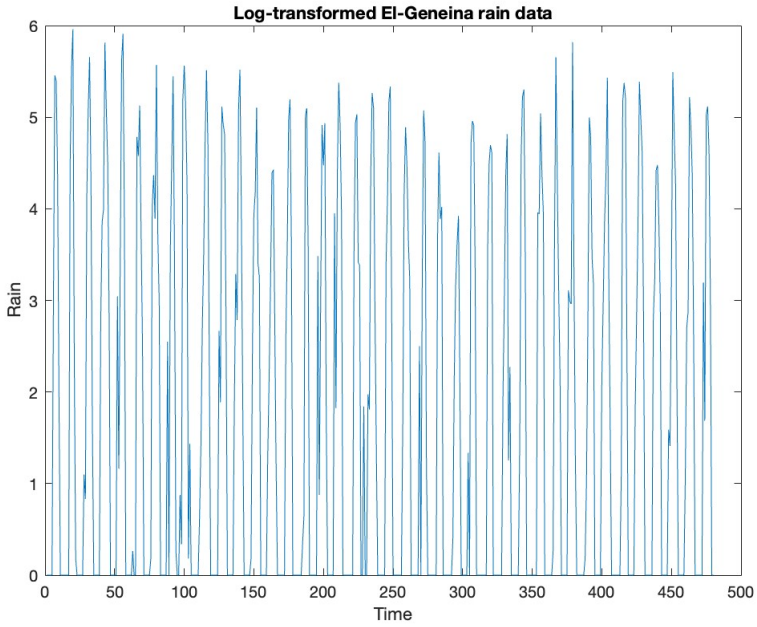


Figure 3: Data before transformation

2.2.2 Trends

Plotting the ACF and PACF of the transformed data showed a clear seasonality of 12 which could be expected since each data point represents one measurement per month. We therefore differentiated the data to remove the periodicities. The differentiated data contained negative values which we interpreted as negative rain which is not realistic. Hence we decided we needed to offset this data. Thus 5 was added to each data point because the lowest value was approximately -4.55775. Using this data and modeling it as an AR(1) process gave a better result than the previous one but the model residuals were not white. The model however had a coefficient $a1 = -0.9672$.

2.3 Reconstruction using Kalman filter

To reconstruct the rain data we used a Kalman filter. The instructions to the project suggested that the measured rain y_t could be described by equation 1.

$$y_t = x_t + x_{t-1} + x_{t-2} + w_t \tag{1}$$

Where x_{t-1} and x_{t-2} are described by equation 2 and 3.

$$x_t = -a_1 x_{t-1} + e_t \tag{2}$$

$$x_{t-1} = -a_1 x_{t-2} + e_t \tag{3}$$

Therefore y_t can be described by equation 4.

$$y_t = -a_1(-a_1 x_{t-2} + e_t) - a_1 x_{t-2} + e_t + x_{t-2} + w_t \tag{4}$$

This was then approximated as equation 5.

$$y_t = a_1^2 x_{t-2} - a_1 x_{t-2} + x_{t-2} \tag{5}$$

This yielded our A-matrix in the Kalman filter shown in equation 6.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & a_1^2 \\ 0 & 0 & -a_1 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

By examining the data of the reconstructed rain we quickly saw that using our A-matrix it rained a bit more for every ten days, which is not that surprising since $a_1 < 0$. We tried using the identity matrix which resulted in the sum of each month being divided by three. In retrospect one could argue that this reconstruction is as good a guess as any really. However, having it rain equally every ten days is not realistic. Furthermore, using data which doesn't fluctuate during the month would not be ideal further down the road. The following matrices, vectors and constants were used for reconstructing the rain with both AR(1) process which are presented in equations 7 through 11.

$$\mathbf{R_e} = 10^{-1} I \tag{7}$$

$$R_w = 0.1 \tag{8}$$

$$\mathbf{R}^{\mathbf{x},\mathbf{x}}_{\mathbf{t}|\mathbf{t-1}} = 10I \tag{9}$$

$$x_{t|t-1} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \tag{10}$$

$$C_{t|t-1} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{11}$$

2.3.1 Reconstructing rain with first AR(1)-model

Using the simple AR(1) process had a difference of the total accumulated rain of 0.2180 units of rain which indicates that that we were on the right track. Plotting the interpolated rain data against the reconstruction is presented below in figure 5.

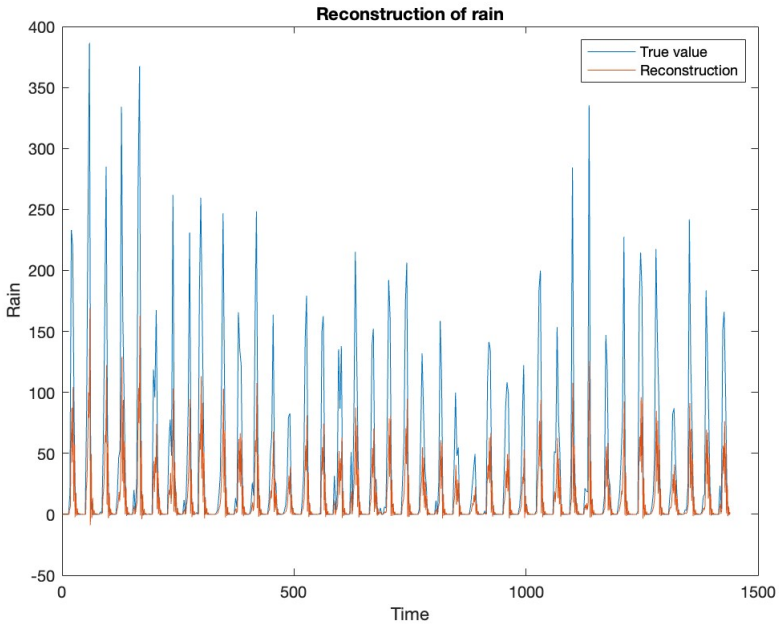


Figure 4: Plot of the reconstructed rain data against the interpolated rain data using the simple AR(1) model.

A reasonable guess would be that approximately a third of the true value (interpolated data) is reconstructed which seems to be the case for most of the data. There are however some discrepancies regarding this which is visible in the plot. In addition to this there is negative rain in the plot which is not ideal which is why we chose to not use this reconstruction.

2.3.2 Reconstructing rain with new AR(1)-model

Using our other model yielded a, in our mind, better reconstruction of the rain. The difference in total accumulated rain was 0.014 units of rain which is also good. Plotting the interpolated rain data against the reconstruction is presented below in figure 4.

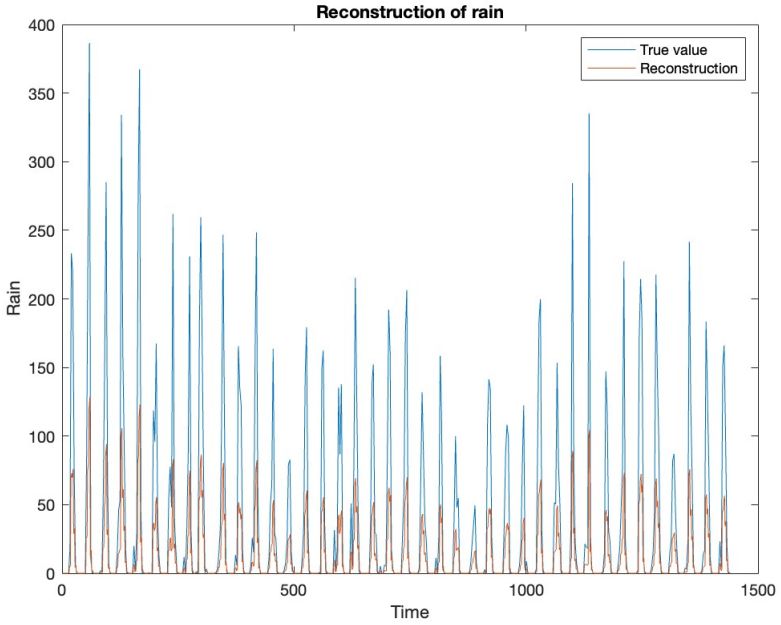


Figure 5: Plot of the reconstructed rain data against the interpolated rain data using the other model.

The distribution of rain seems better compared to the previous reconstruction and it doesn't contain any negative rain which we thought was better.

3 Modeling and validation of El-Geneina

3.1 Data transformation

To model the nvdi data we first needed to transform the data since the data was stored as integers ranging 0 to 255. These needed to be rescaled to $[-1, 1]$. This was done by first dividing every data point by 255 and then multiplying that data point with 2 and lastly subtracting 1. Unlike in rain data we chose not to use a power transformation because this resulted in problems further down the road.

3.2 Trends

Plotting the ACF and PACF for the rescaled NVDI data showed a periodicity of 36 which is shown in figure 6.

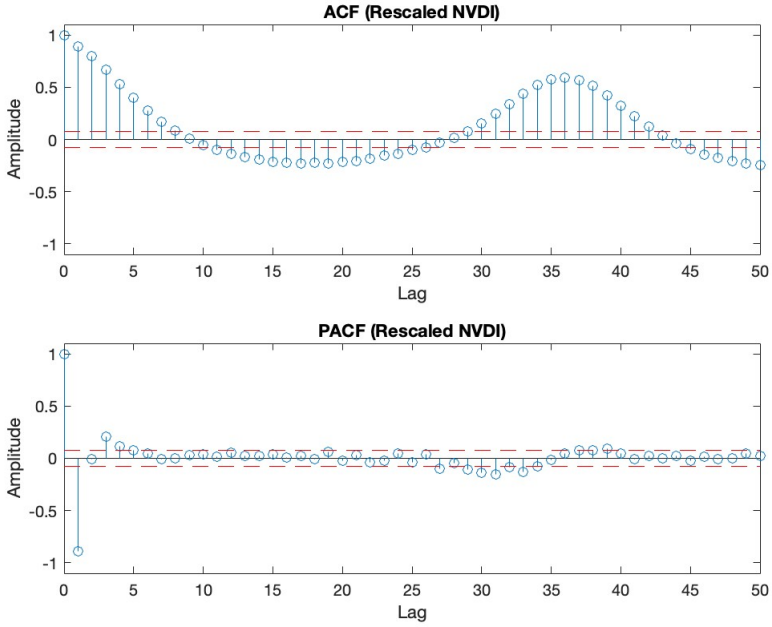


Figure 6: ACF and PACF for rescaled NVDI data.

This periodicity was not any surprise since there are 36 measurements each year, i.e. a yearly seasonality. To handle this the whole data set was differentiated with a season of 36 which is described by equation x.

$$\nabla_{36} y_t = (1 - z^{-36}) y_t \tag{12}$$

Plotting the ACF and PACF for the seasoned data has a lower dependency of the yearly season which can be seen in figure 7.

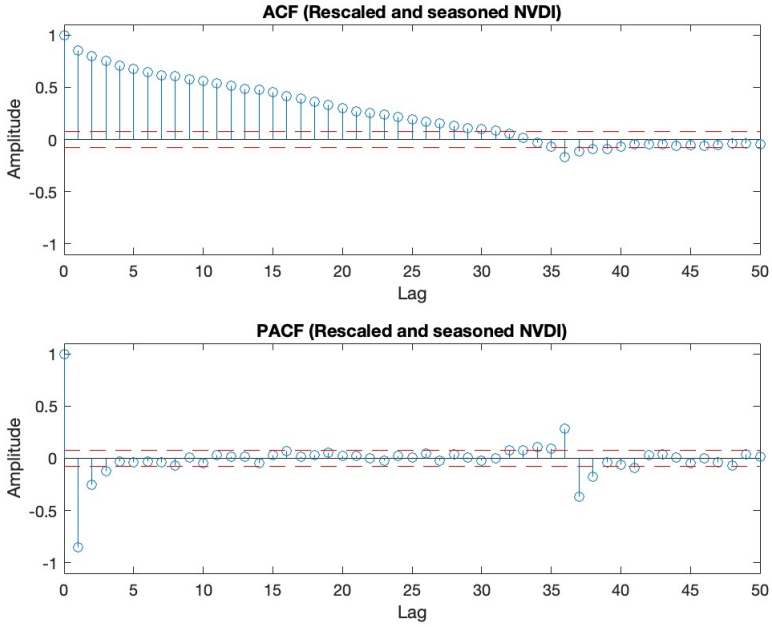


Figure 7: Plot of ACF and PACF for rescaled and seasoned NVDI data.

3.3 Splitting

To evaluate our model we had to split our data into training data and validation data while also leaving some data for test data. Our whole data set contained 648 data points of which 450 were selected to be training data, 135 data points validation data and the remaining 63 data points were left out for test data.

3.4 Modeling

After the split we once again plotted the ACF and PACF in order to create a model. After trying many different models and checking the residuals as well as the whiteness of the training and validation data we ended up with an ARMA(1,37) model which is presented in table 1.

$A(z) = 1 - 0.9483(\pm 0.0174)z^{-1}$
$C(z) = 1 - 0.2286(\pm 0.05215)z^{-1} - 0.9097(\pm 0.02701)z^{-36} + 0.3183(\pm 0.04957)z^{-37}$

Table 1: Polynomials in ARMA(1,37) model.

We tried simpler and more complex models but to abide with the KISS-rule (Keep it simple stupid) we were satisfied enough with the result while still not having too many parameters. Worth noting is that all parameters can be deemed significant based on the coefficients and the confidence intervals. The ,model received 21.27 on training data and 20.57 on validation data.

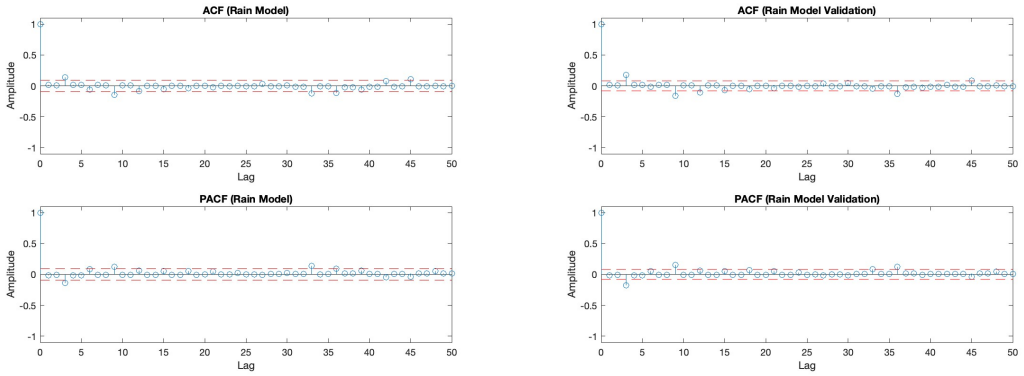
3.5 Box-Jenkins Model

3.5.1 Modeling the input data

When solving this task we used the framework provided by lab 2 to create a Box-Jenksins model. Since rain data was acting as input signal to the model we want to pre-whiten it and find a suitable model for it. This is done on the rain data created in part A of the project. This was made doing regular residual analysis looking at the ACF and PACF of the residual. Here we found a model that had a white residual for the training data which later yielded a white residual on the validation data. The final model we used for the input data was an ARMA(37,3). All of the parameters in the model were significant and their values with included confidence intervals are present below in table 2 as well as the ACF and PACF for the model in figure 8. This gave a result of 23.65 on the Monti-test for the residual of the training data and 24.88 on the residual of the validation data of the rain.

$A(z) = 1 - 0.9198(\pm 0.02352)z^{-1} - 0.7179(\pm 0.03674)z^{-36} + 0.6451(\pm 0.04225)z^{-37}$
$C(z) = 1 - 0.7146(\pm 0.04216)z^{-3}$

Table 2: Polynomials in ARMA(37,3) model.



(a) Training data (b) Validation data

Figure 8: Plot of ACF and PACF for rain data in El-Geneina.

3.5.2 Cross-Correlation

The next step when creating the Box-Jenkins model was to determine the constants d, r and s. Using the A3 and C3 polynomials we can extract the white noise, w_t, and ϵ_t by filtering with our rain data as well as our nvdi data. Now we look at the cross-correlation between w_t and ϵ_t to estimate the orders of our Box-Jenkins model. The cross-correlation plot can be seen in figure 9.

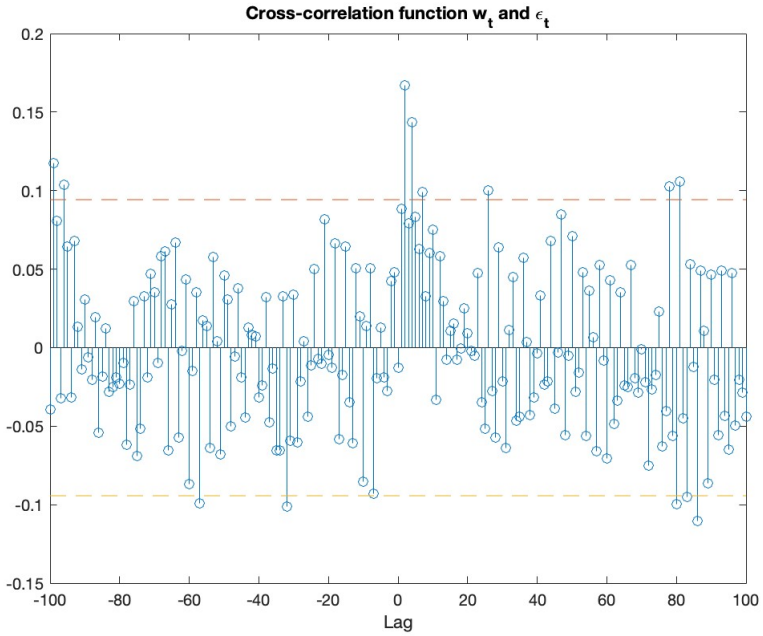


Figure 9: Plot of the cross-correlation between w_t and ϵ_t.

According to theory one should be able to pick d, r and s quite simply but we found it very difficult if we stuck to what theory would have us pick. After discussing it with Andreas Jakobsson and having in mind that our data is not in fact gaussian (which theory assumes it to be) we once again abided by the KISS-rule which resulted in us choosing $d = 2$, $r = 0$ and $s = 0$. At first we tried other values of d, r and s with the ambition of having $\tilde{e}$ and the rain uncorrelated. This however proved impossible and with the knowledge that our data is not gaussian we knew that we coudln't trust confidence intervals and thus had to turn a blind eye with regards to the data not being completely uncorrelated.

We then got the model for our Mba2 which was a very simple model. In fact it only had the B polynomial $B(z) = 0.004672(\pm 0.0007684)z^{-2}$.

3.5.3 Extracting $\tilde{e}_t$ using our estimated orders

Having chosen the orders of the A2 polynomial, the delay in the B polynomial and the order of the B polynomial we extract the $\tilde{e}_t$ process as the residual when using the found polynomials with our rain and NVDI. With $\tilde{e}_t$ we look at the cross-correlation between x and $\tilde{e}_t$ in figure 10, which should be uncorrelated as stated above.

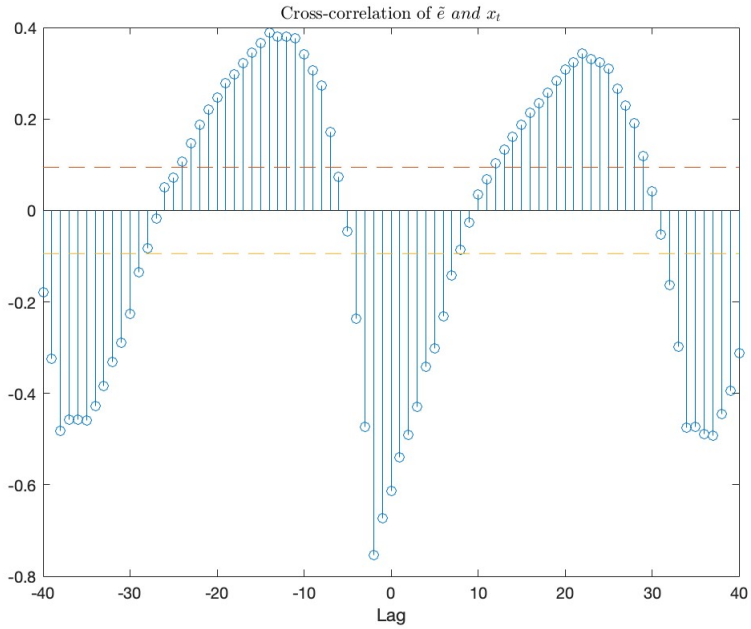


Figure 10: Plot of the cross-correlation between $\tilde{e}_t$ and x_t.

As we can see in figure 10 they are not exactly uncorrelated but it did not work to get both uncorrelated $\tilde{e}_t$ and x_t as well as a working model. So we made the choice of looking at this plot with an open mind.

3.5.4 Finding model for $\tilde{e}_t$

The final process for $\tilde{e}_t$ was modeled as ARMA(36,36) with the corresponding parameters which are presented in table 3 below. Note how all parameters are significant and how few of them there are. The ACF and PACF are also presented in figure 11 below and everything looks good. The residual got a whopping 12.91 on the Monti-test, which we are very satisfied with.

$A(z) = 1 - 0.7484(\pm 0.03236)z^{-1} - 0.209(\pm 0.0353)z^{-36}$
$C(z) = 1 + 0.2458(\pm 0.05694)z^{-36}$

Table 3: Polynomials in ARMA(36,36) model.

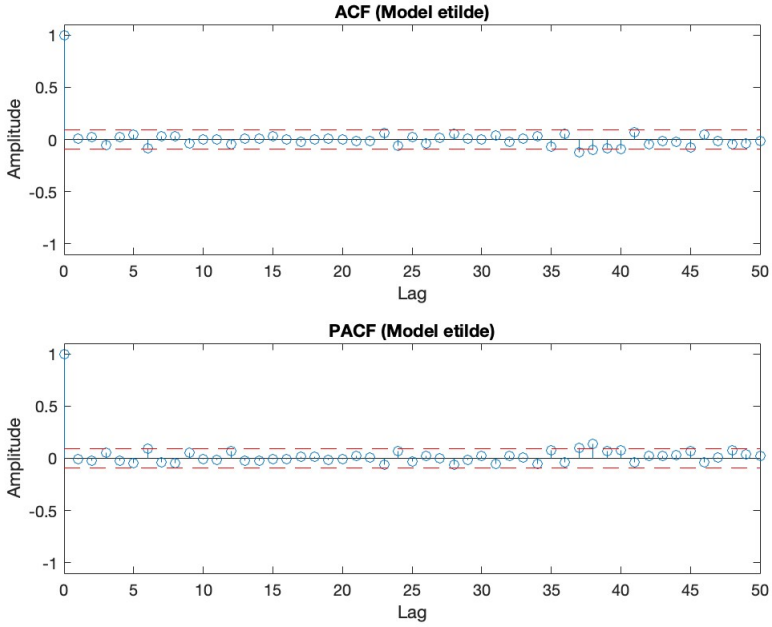


Figure 11: Plot of the ACF and PACF for the residual of $\tilde{e}_t$.

3.5.5 Creating our final Box-Jenkins model

A note before digging into our final creation of the time independent Box-Jenkins model: Our process above was done only on the training data without touching the validation data as usual. So the reason for doing all of the above is not only to find the orders of our Box-Jenkins models polynomials but also to use our values found as initial predictions for what our parameters in our polynomials will be equal to.

Our C and D polynomials are both of order 36 we therefore chose to free the model with the same model orders in our Box Jenkins as we had in our "guessed" polynomials. This gave us our final Box-Jenkins model with parameters presented in the table 4 below. As we can see all of our parameters are significant even though our B polynomial has extremely low values it is vital since that is the one which interacts with our rain. Furthermore our rain data has much larger values than the vegetation data which makes it more "impactful" than what one might intitally think.

$B(z) = 0.00453(\pm 0.0002488)z^{-2}$
$C(z) = 1 + 0.2736(\pm 0.05726)z^{-36}$
$D(z) = 1 - 0.8606(\pm 0.03168)z^{-1} - 0.1513(\pm 0.03292)z^{-36}$

Table 4: Polynomials in the final Box-Jenkins model.

3.5.6 Residual analysis of Box-Jenkins

Now we look at the residual from the Box Jenkins model with respect to the rain data and vegetation-data. As we can see the residual from our Box-Jenkins looks good in the ACF and PACF, can be seen in figure 12 below. The residual also got a 17.66 on the Monti-test indicating that our model is good for our training data.

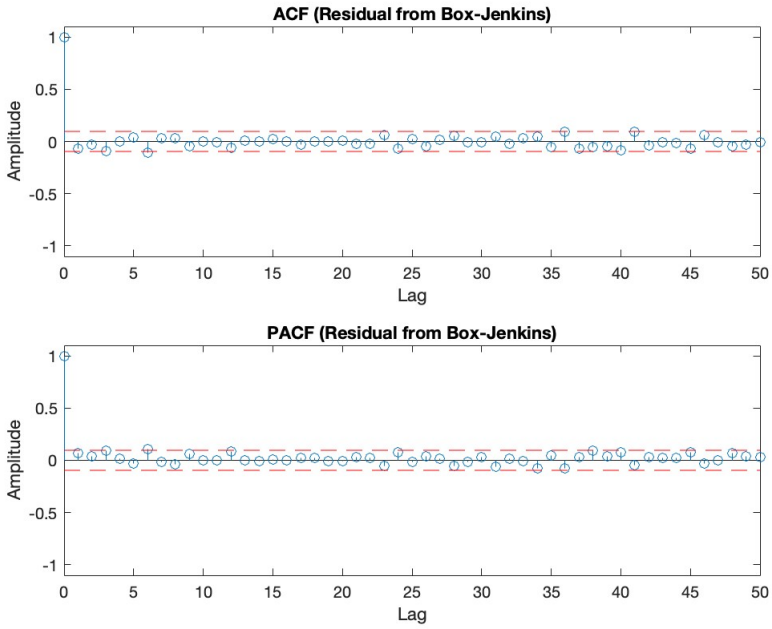


Figure 12: Plot of the ACF and PACF for the residual of our Box-Jenkins model.

Preferably we would also like to see no correlation between our residual and our input. Which is why we plotted the cross-correlation between the two below in figure 13, as we can see there are staples 5 sticking out of the confidence intervals however on the positive x-axis and only one of them are doing so significantly and again we are not to trust the confidence intervals entirely.

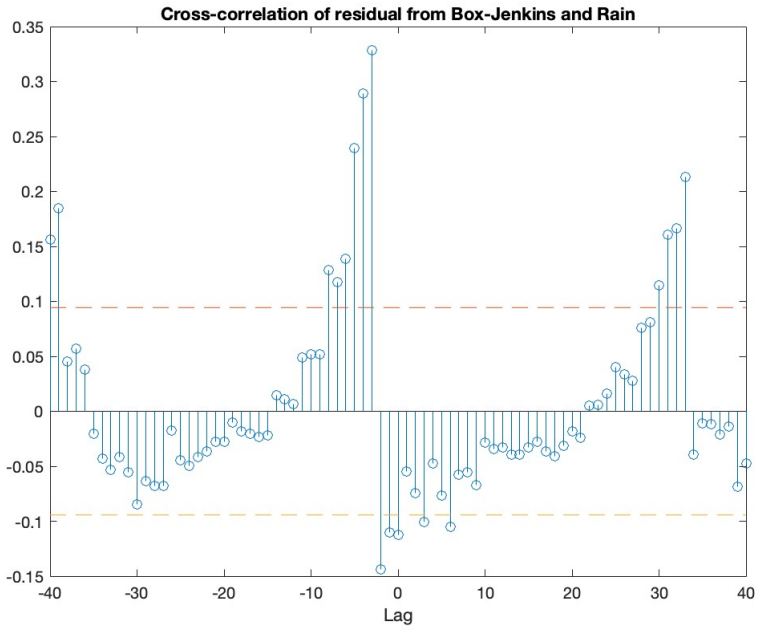


Figure 13: Plot of cross-correlation between the residual of our Box-Jenkins model and input on training data.

Moving on to testing our model from the training data on the validation set which resulted in results getting 26.62 on the Monti-test for the residual and an ACF and PACF presented in figure 14 below. As you can see

it is not perfect but it looks rather good.

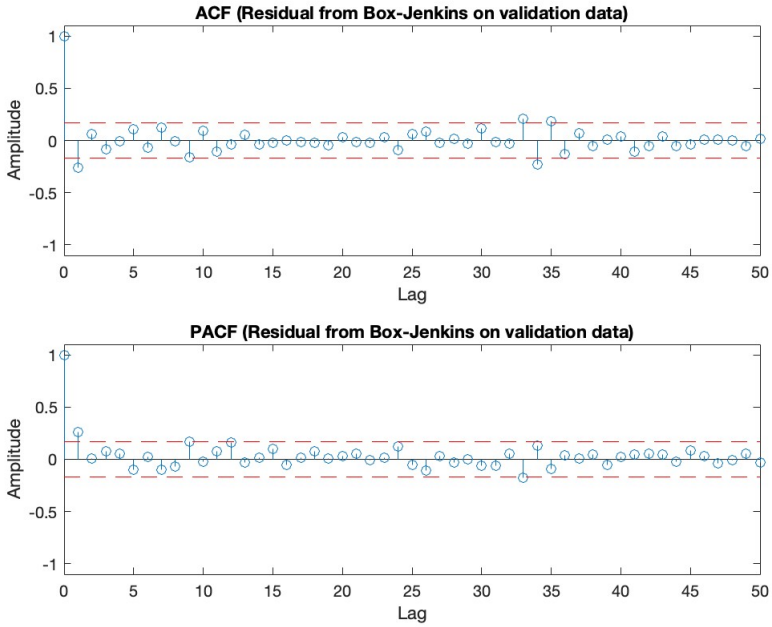


Figure 14: Plot of the ACF and PACF for the residual of our Box-Jenkins model on the validation data.

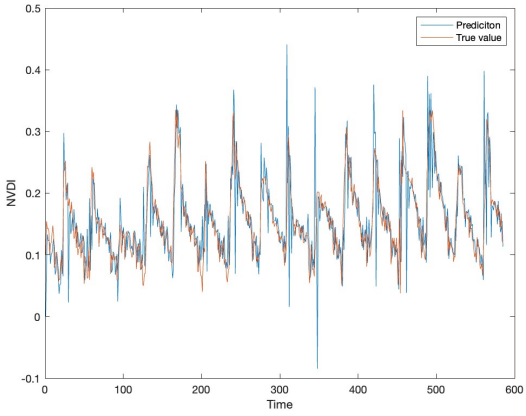
3.5.7 Predictions with Box-Jenkins

Now we are going to use our found Box-Jenkins model in order to predict the vegetation based on the rain. In order to do this we also have to predict the rain since we need the future rain to predict future vegetation. In order to predict the future rain without using time varying models we instead of using Kalman filters we predict the rain using two lines of code which we got from the lectures and is presented below.

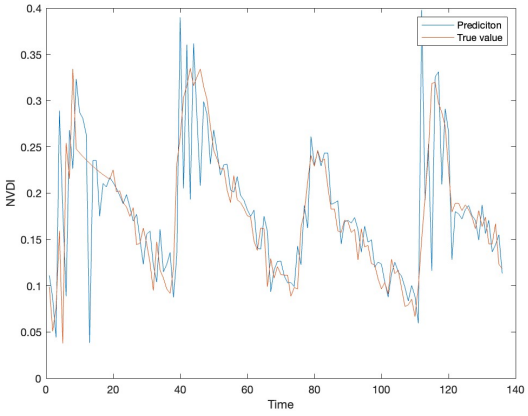
```
[Fx,Gx] = polydiv(Rain_model.c,Rain_model.a, k );
xhatk = filter(Gx,Rain_model.c,x);
```

When forming the prediction for the vegetation we did it the same way we did in lab 2 during the course in which we predicted using our Box-Jenkins. This was done convoluting our polynomials from our Box-Jenkins and solving the Diophantine equation.

The results we got for the 1-step prediction were pretty good, the residual was white for both the training and the validation data indicating a good model. The results on the Monti-test were 15.02 on the training set and 26.62 on the validation set. Figure 15 of the 1-step predictions compared to the original vegetation are presented for both the training set and the validation set but also for only the validation set.



(a) 1-step prediction on NVDI training and validation data.

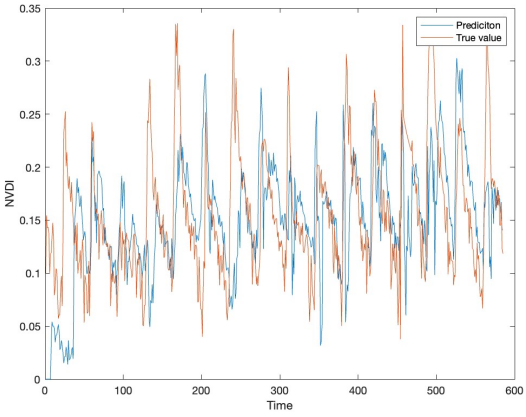


(b) 1-step prediction on NVDI validation data.

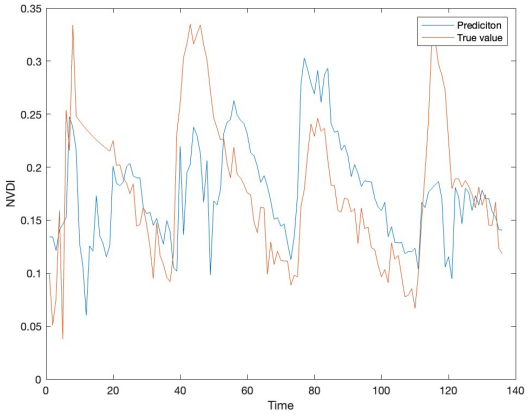
Figure 15: Plots of prediction on NVDI data and with Box-Jenkins.

As we can see the Box-Jenkins does a good job at predicting the rain however as we can see it might "overshoot" some when the vegetation starts increasing or decreasing. As we can see in the first part of the validation data we have a big difference between the prediction and the actual value. But if you look closer at the vegetation dataset it is clear that those points are interpolated, it might be some of the missing data points in the set that were mentioned in the project description. This means our actual 1 step predictor might actually be better for approximating these points.

Now to 7-step prediction, which to no surprise does not work as well as our 1-step predictor. When looking at the residual of the 7-step predictor we see that it is not white, not really that close either, it got around 135 at the Monti-test. But this is not that bad, if we look at figure 16 below of the 7-step for both the whole dataset and just for the validation set we can see that our predictor has the same look but it is not as spot on as the 1-step prediction. When we look at the validation data it actually looks pretty good, it follows the same patterns but does not manage to capture the magnitude of the peaks perfectly.



(a) 7-step prediction on NVDI training and validation data.



(b) 7-step prediction on NVDI validation data.

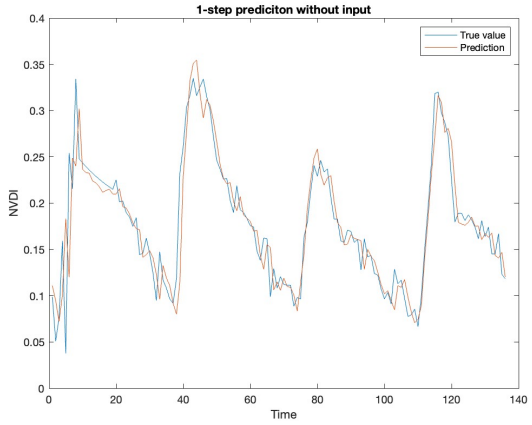
Figure 16: Plots of prediction on NVDI data and with Box-Jenkins.

3.6 Predicting the vegetation without input and without time-varying coefficients

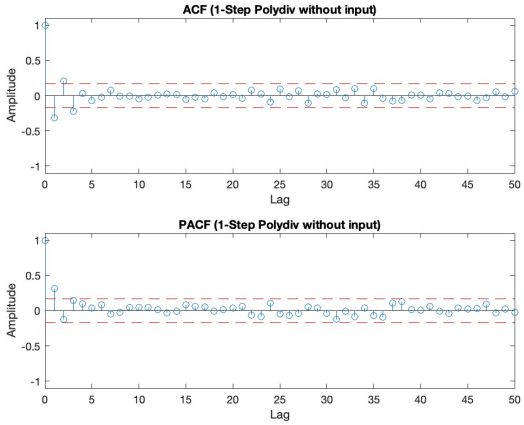
To compare our Box-Jenkins predictor with an external input (rain) we predicted the vegetation in the same manner as our input in the Box-Jenkins was predicted, i.e. using the following code:

```
Aconv = conv(AS,model_ar1ma37.a); % AS = [1 zeros(1,35) -1];
[~, Gx] = polydiv( model_ar1ma37.c, Aconv, k );
yhatk = filter(Gx, model_ar1ma37.c, scaled_nvdi(1:585));
```

1-step and 7-step predictions along with their corresponding ACF and PACF are presented in figure 17 and 18. The predictions are really great especially for the 7-step compared to our other stationary problems. The result on the Monti-test for the residuals of the validation data was 25 for the 1-step and 90.53 for the 7 step. When doing this prediction we used our polynomials from the vegetation model as our input in the filter and polydiv.

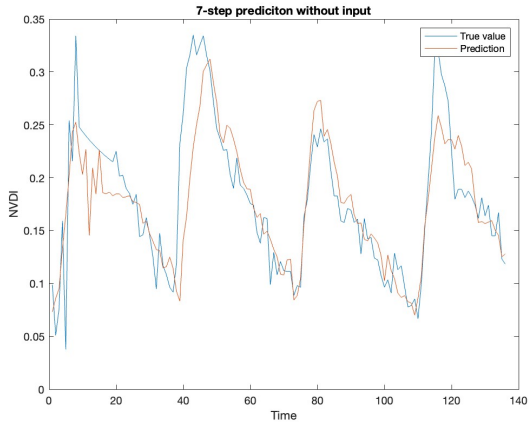


(a) Plot of 1-step prediction.

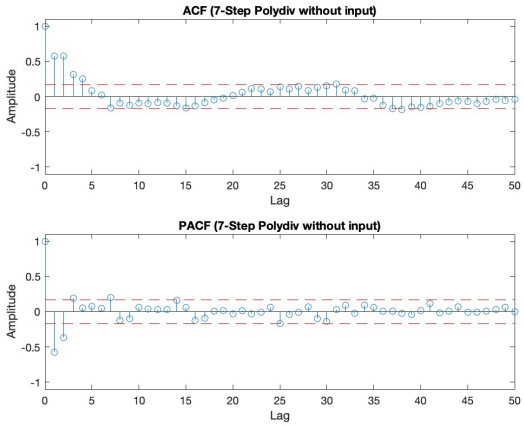


(b) Plot of ACF and PACF.

Figure 17: Plots regarding 1-step prediction on NVDI without input and without time-varying coefficients.



(a) Plot of 7-step prediction.



(b) Plot of ACF and PACF.

Figure 18: Plots regarding 7-step prediction on NVDI without input and without time-varying coefficients.

4 Time-varying model for El-Geneina

4.1 Kalman filter without input

We are aware that the task is to update the model with input but we thought it would be useful for comparison. To exclude the differentiation which we did to obtain our model in the previous task we convoluted our A-polynomial with our season. To recursively update our model we implemented a Kalman filter in accordance with lab 3 as well as code examples provided in the course. The filter was implemented for the 1-step and 7-step prediction with the following matrices, vectors and constants which are described by equations 13 through 18.

$$\mathbf{A} = I \tag{13}$$

$$\mathbf{R_e} = 10^{-5} I \tag{14}$$

$$R_w = 0.1 \tag{15}$$

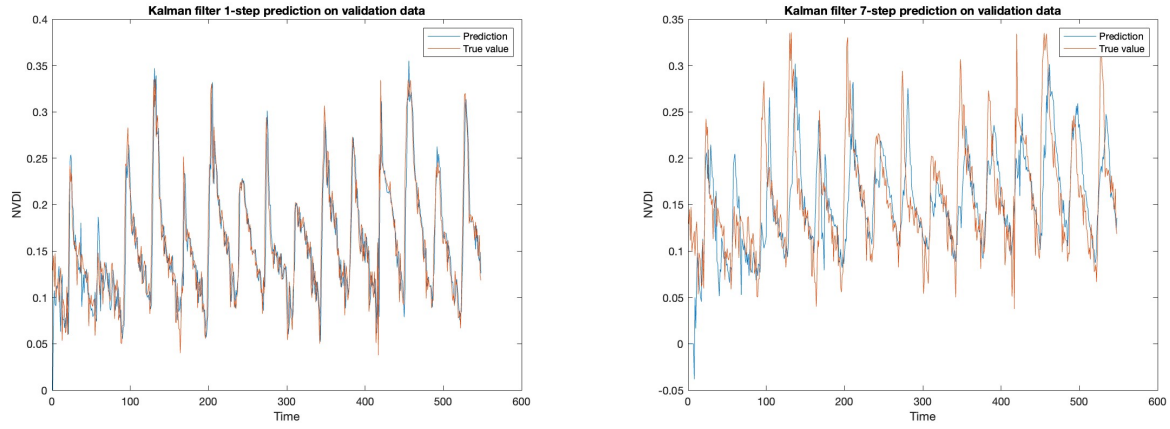
$$\mathbf{R^{x,x}_{t|t-1}} = 10I \tag{16}$$

$$x_{t|t-1} = \begin{bmatrix} a_1 & a_{36} & a_{37} & c_1 & c_{36} & c_{37} \end{bmatrix} \tag{17}$$

$$C_{t|t-1} = \begin{bmatrix} -y_{t-1} & -y_{t-36} & -y_{t-37} & \hat{e}_{t-1} & \hat{e}_{t-36} & \hat{e}_{t-37} \end{bmatrix} \tag{18}$$

When creating our observation vector $C_{t|t-1}$ we base this on the model for El Generina, and our initial states are the values of our a and c coefficients. But they will of course be updated in the filter recursively

The results on the Monti-test for 1-step and 7-step prediction were: 22.26 on validation data and 13.35 on the training data for 1 step. And for 7-step 119 on the validation data. And as we can see the prediction is accurate for the 1 step over the whole data set including the validation data. The 7-step however struggles a bit more but is still good, we can see that it clearly does a good job at predicting trends but it is somewhat delayed when we have strong and narrow peaks. This shown in figure 19.



(a) Plot of 1-step prediction.

(b) Plot of 7-step prediction.

Figure 19: Plots of prediction of NVDI validation data with Kalman filter without input.

Now if we look at the states we are letting vary over time and see how they change we can determine if they are significant or not we can see that none of them converges towards zero nor are zero at any time indicating that they are significant and should not be removed. This can be seen in the figure 20 below of the six different states over time plotted against their initial values.

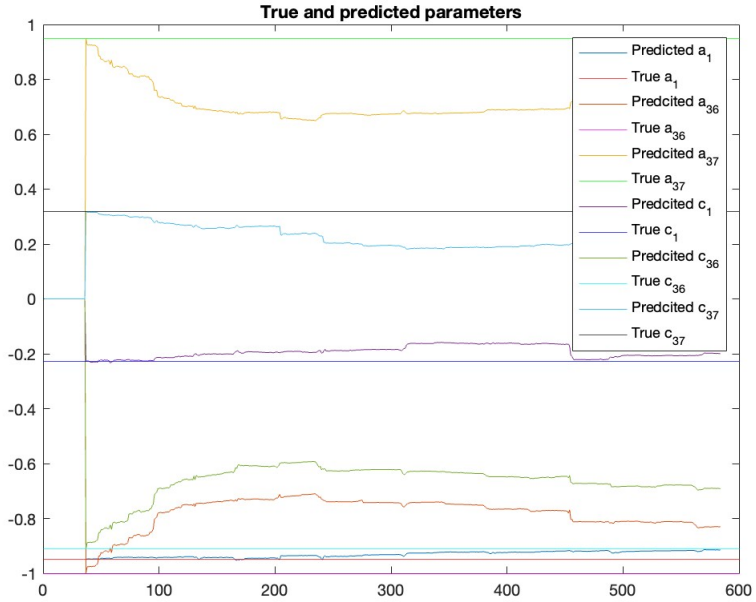


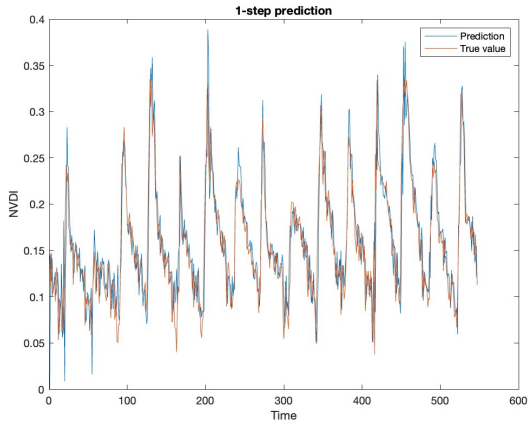
Figure 20: Plot predicted and true values of parameters in Kalman filter without input.

4.2 Kalman filter with input

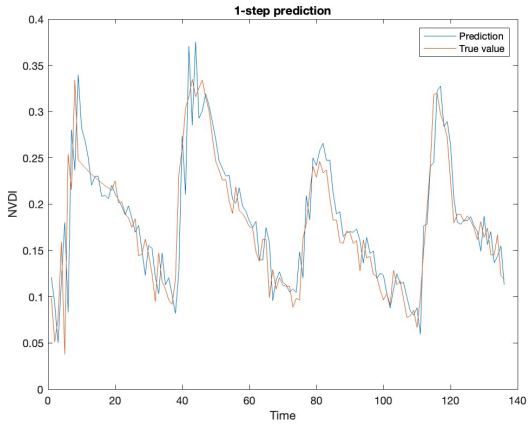
Now to the final boss where we use a Box Jenkins model combined with a Kalman filter to make a k-step prediction. In order to create the filter we set our observation matrix and our initial states based on our Box Jenkins model calculated in B since that model worked rather well. Since we now have an input in our Kalman filter in the form of rain, we have to take this into consideration when making predictions. Luckily for us our model for the vegetation is not affected by the rain in the same period but there is a delay of 2 time-steps i.e 20 days which is reasonable. This means that for a k-step prediction for the vegetation we only have to make a (k-2)-step prediction of the rain. One way to do this is to have a Kalman filter for the rain inside the Kalman filter for the vegetation, this seemed like a good idea until we tried and realized we do not have the programming skills to do it. Instead we predicted the rain the same way we did in the Box Jenkins model, using polydiv. And I think we got kind of creative here since for each of the predictions of the rain, only using the rain up until time t for each timestep and we also got negative rain sometimes when using polydiv. In order to prevent the negative rain we just had an easy solution which worked fine, this was just setting all the negative rain to zero. An example is presented below for the last step in our predictor.

```
[~, Gx] = polydiv( model_ar37ma3.c, model_ar37ma3.a, 5 );
xhat5 = filter(Gx, model_ar37ma3.c, x(1:t));
    if xhat5(t) < 0
        xhat5(t) = 0;
    end
 Ct7 = [yhat6 y(t-29) xhat5(t) xhat4(t) x(t-31) ehat(t-29)];
 yhat7 = Ct7*x_t1;
```

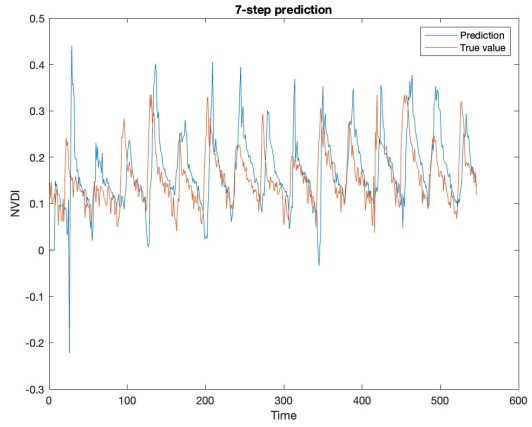
Now if we plot the 1-step and the 7-step predictor using our Kalman filter we can see the results in the figure 21 below. As we can see our 7-step prediction works alright but not great, the residual gets a score of 133.27 on the Monti-test for the validation data. We can see that our predictor does a decent job but clearly can't handle what we think is linear-interpolated data in the data points between 10-25 in the validation data set.



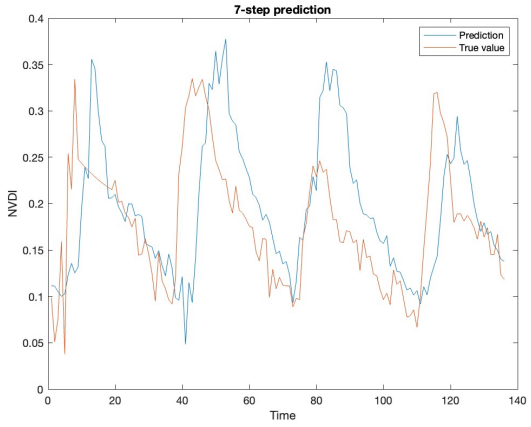
(a) 1-step prediction on training and validation data



(b) 1-step prediction on validation data



(c) 7-step prediction on training and validation data

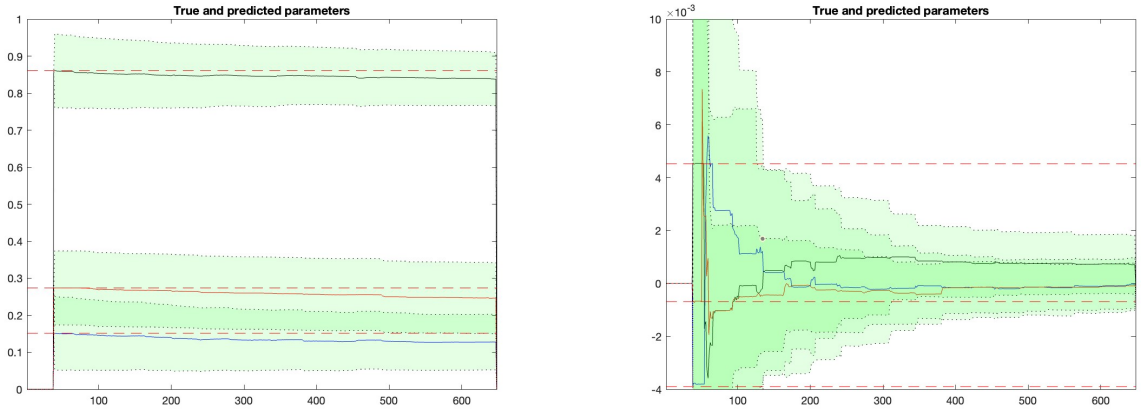


(d) 7-step prediction on validation data

Figure 21: Plots of prediction using Kalman filter with input.

If we look at the 1-step prediction however it works pretty good as you can see in the figure 21. It is doing a decent job at predicting and is almost aligned with the validation part of the realization data. It scored 34.63 on the Monti-test for the validation part, hence not deemed white. However if you look at the graph it looks pretty good and we are happy with it. And as stated before our model does not handle outliers so when we for example don't use the first 5 points of the validation test when looking at the residual the result on the Monti-test is quite different. This instead gives a white residual with a Monti-test score of 26.35. We know that this is not a legitimate result, but the large difference on the Monti-test indicates that those particular data points significantly worsen the prediction. Because of this we thought it would be worth mentioning in the report. Also worth mentioning is that we lowered R_e from 10^{-6} to 10^{-9} because iwe have rather stationairy data.

If we look at the states in figure 22 some of them remain constant and non-zero over time and some vary a lot. The ones that vary are the ones that are multiplied with present rain data. This may be due to inconsistency in our rain data and it not being modeled perfectly. As we can see the ones connected to the rain are small and vary a lot with a high variance while the others change slowly and are close to our initial values. Now for the last part I will try to remove some parameters, which might be tricky since we already have so few.



(a) KA(1) (black), KA(36) (blue) and KC(36) (red). (b) KB(1) (black), KB(2) (blue) and KB(37) (red).

Figure 22: Plots of predicted and true values of parameters in Kalman filter with input .

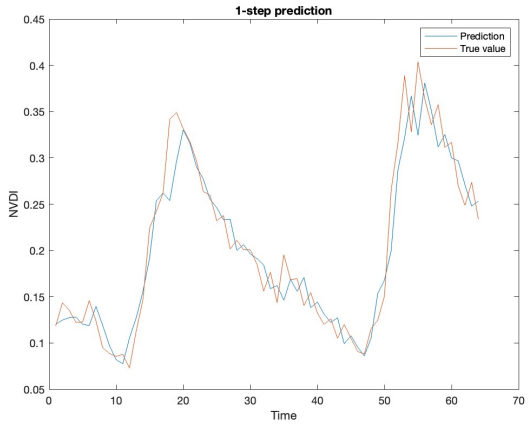
At first we tried removing all parameters except the x_{t-2} parameter. The prediction got a little worse but it barely seemed to make a difference. This led us to removing that parameter as well which means that this Kalman filter is without input once again, however with fewer parameters than the filter without input described earlier, three to be specific. It was not exactly surprising that the model worked worse than our other Kalman filter without input which had 6 parameters hence we will not include this model. But what was interesting was that it worked as well on the validation data as our Kalman with rain. Therefor removing any parameters does not make sense if we want to use the rain as input it makes model worse without making it that much simpler.

It is noteworthy that it performs worse than our original Kalman filter without rain as input but marginally so. It seems as if using the rain as input does not improve the model but rather cripples its capability to predict when using models of similar complexity.

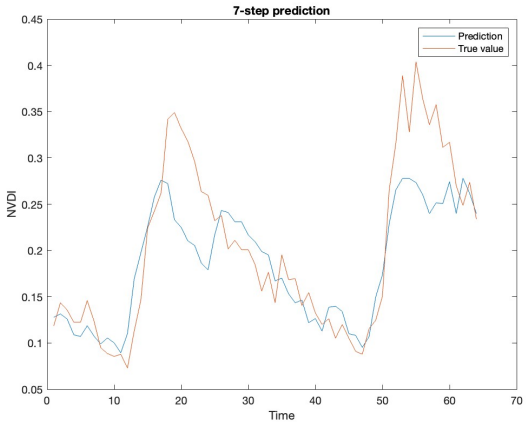
5 Testing models on test data

So now we want to try our models on the test data and see how well they perform, how fun! We used ten percent of our total data set as our test data and this led to our test data being 64 data points which is almost 2 years. Now to some testing.

When we try our best model, polydiv without input, on the test data we get great results. On the 1-step and 7-step we get 20.24 and 64.05 on the Monti-test and the plots of our k-steps are presented below in figure 23, it looks really good right? This model working this good on the test data means we have a working model, at least for 1-step. We can't expect our model to work perfectly in 7-step predictions.



(a) Plot of 1-step prediction on test data.

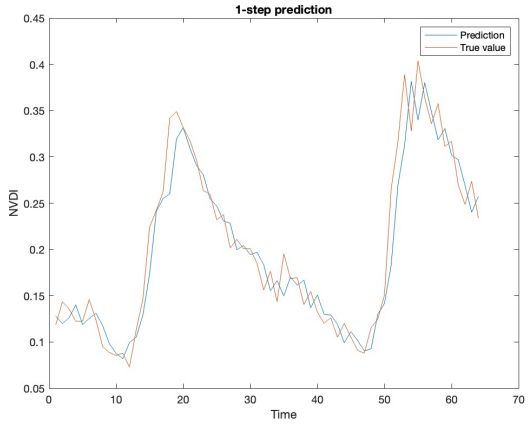


(b) 7-step prediction on test data.

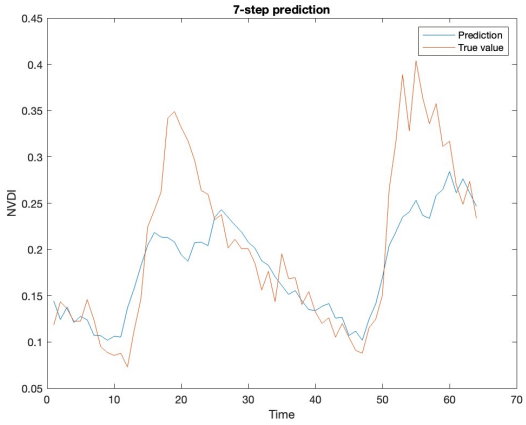
Figure 23: Plots of prediction on test data using polydiv without input.

Now the Kalman filter without input which can be seen in figure 24, it does look pretty similar to the stationary with no input might even be slightly worse which is not very reassuring since our states are being updated to make a better prediction.

22.01 on the Monti-test for the 1-step and 71.64 and the figures can be seen below. The reason why they look so similar is due to our states not changing that much.



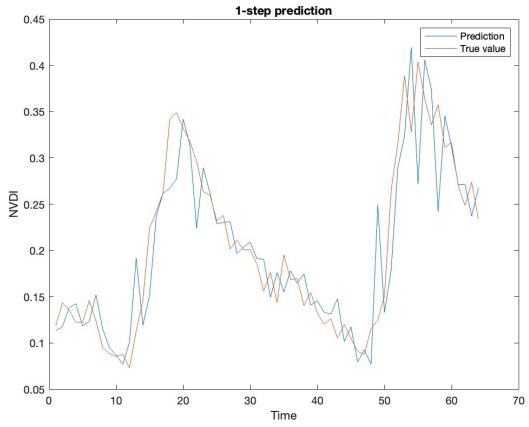
(a) Plot of 1-step prediction on test data.



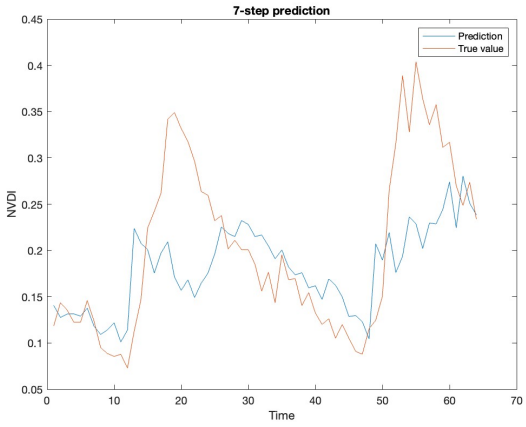
(b) 7-step prediction on test data.

Figure 24: Plots of prediction on test data using a Kalman filter without input.

Test data prediction for Box-Jenkins predictor which is visible in figure 25, as we can see the 1-step does a good job at predicting and looks pretty good, the Monti-test only resulted in a score of 27.40 while the 7-step predictor got 66.16 on the Monti-test when the 7-prediction clearly works a lot worse than our other 7-step predictors. Hence we can not really trust the Monti-test at all times but also have to inspect the graphs to see.



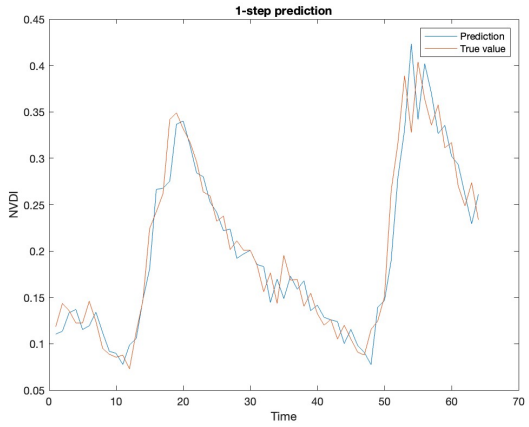
(a) Plot of 1-step prediction on test data.



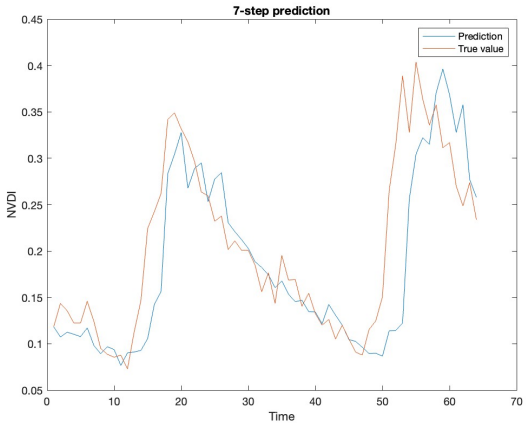
(b) 7-step prediction on test data.

Figure 25: Plots of prediction on test data using Box-Jenkins.

Test data on Kalman with input, with on Monti-test (25,58) the prediction looks good which is visible in figure 26, we really like it, looks very smooth. Now if we look at the 7-step prediction on the test data it actually is our best prediction yet on the test data, which indicates that maybe we did not do all this in vain. Very exciting to see some results from our final model



(a) Plot of 1-step prediction on test data.



(b) 7-step prediction on test data.

Figure 26: Plots of prediction on test data using a Kalman filter with input.

Now to make a numerical evaluation of how well our model works on our validation and our test data set. This is done by calculating the normalized variance of our residuals as well as the sum of mean squared errors of our residuals. This is presented in table 5 below.

Model	Normalized variance	MSE
Polydiv without input (1-step)(Val)	0.1801	8.2110e-04
Polydiv without input (7-step)(Val)	0.3123	0.0014
Polydiv without input (1-step)(Test)	0.0894	7.4451e-04
Polydiv without input (7-step)(Test)	0.2677	0.0025
Kalman without input (1-step)(Val)	0.2227	7.6650e-04
Kalman without input (7-step)(Val)	0.5640	0.0026
Kalman without input (1-step)(Test)	0.0906	7.6650e-04
Kalman without input (7-step)(Test)	0.3575	0.0034
Box Jenkins (1-step) (Val)	0.5989	0.0027
Box Jenkins (7-step) (Val)	1.0447	0.0048
Box Jenkins (1-step) (Test)	0.2262	0.0137
Box Jenkins (7-step) (Test)	0.6432	0.0057
Kalman with input(1-step) (Val)	0.2479	0.0011
Kalman with input(7-step) (Val)	1.1799	0.0054
Kalman with input(1-step) (Test)	0.0977	8.0970e-04
Kalman with input(7-step) (Test)	0.4295	0.0038

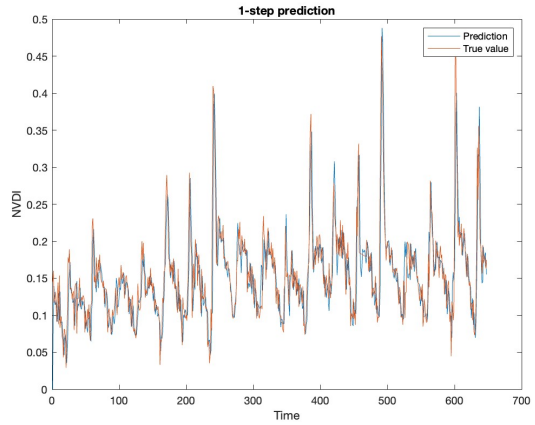
Table 5: The different models with their corresponding normalized variance and mean squared error.

So how do we interpret this table? Well a low MSE on the test data should be priority one because this indicates that our model actually works on a data set which we have never seen before. And a low MSE with high normalized variance indicates that our prediction actually works really good most of the time and really bad a few times. An explanation for this is not handling "outliers" in the data set and our prediction which is fine otherwise becomes really bad. A low MSE with a low variance would mean our predictor is slightly off a lot of the time. Depending on what we want our model to do it seems that this is the best option when measuring vegetation due to how important it is to the local population. Being extremely wrong could mean dire consequences. The numerical values in table 10 are talked about in the discussion.

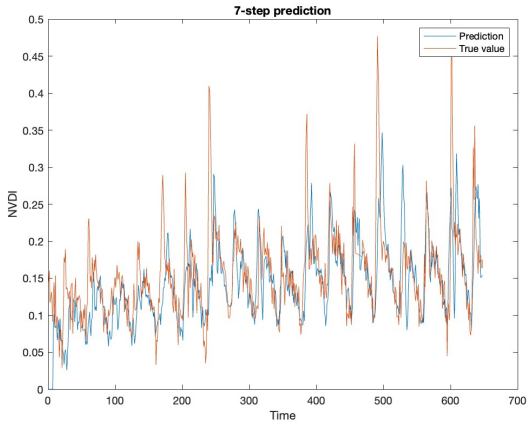
6 Modeling for Kassala

When using our models from El-Geneina on the Kassala data we first have to recreate the rain from Kassala. This is done the exact same way as earlier which had the same result as the reconstructed rain for El-Geneina.

When that was done we could try our models for Kassala. Testing the polydiv without input for 1-step prediction on Kassalas. Here of course we look at the entire data set and look at the result. So for the 1-step-prediction the results actually looked pretty good, not great. The prediction is presented in figure 27 below. But if we look at the numbers we can see that the normalized variance is 0.3175 and the MSE is 0.0011 which is decent for a 1-step predictor, worse than on our test data but not bad. I was not intending to use the 7-step predictor with this model but could not help myself; this can be seen in the figure 27. It is actually not bad at all,the normalized variance is 0.802 and MSE 0.0029 which is only a little bit worse than our 7-step prediction on our test data which seems odd since the 1-step predictor differed more.



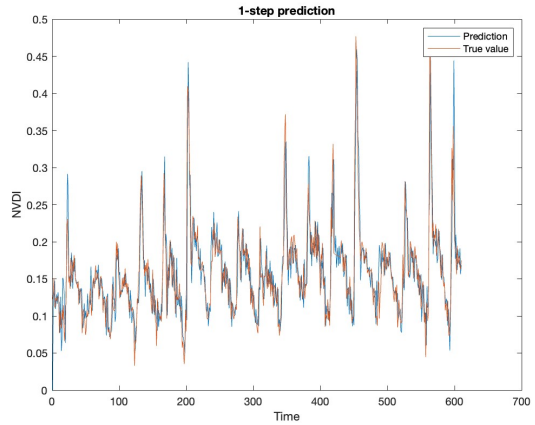
(a) Plot of 1-step prediction.



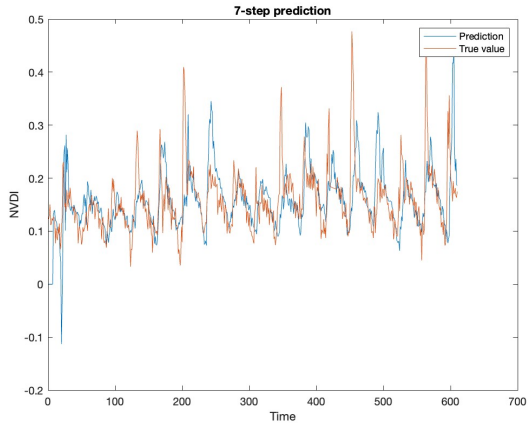
(b) 7-step prediction.

Figure 27: Plots of prediction on test data using polydiv on Kassala data without rain as input.

Now we try using our Kalman filter with Kassala data, with the reconstructed rain for Kassala. The choice of using this model even though it got worse numerical results is pointed out in the discussion. The figures of the 1-step and 7-step prediction are presented below in figure 28. For the 7-step it looks pretty good/half decent the MSE is 0.0042, exactly the same as the MSE from our test-data 7-step in ElGeneina which is interesting. However the normalized variance is a lot higher than for El-Geneina; it was 1.2183 here. And as you can see the 7-step prediction does in fact look worse but not much worse. Now to the 1-step predictor which we can see works very well, not as well as on El-Geneina but pretty good still satisfyingly well. The MSE was again worse than El-Geneina, 0.0011 but not that much worse as one could expect and the variance of the residual was in fact a lot higher (0.3075), three times as much indicating that our predictor does go way more off for Kassala at times than compared to El-Geneina. As we now by now a low normalized variance for the residual makes better models.



(a) Plot of 1-step prediction.



(b) 7-step prediction.

Figure 28: Plots of prediction on test data using a Kalman filter with input on Kassala data.

Discussion on why the results in one-step differ a lot more than our results in 7-step for Kassala and El-Geneina, our reasoning here would be that our models are so much worse for the 1-step when trying them on Kassala since the climate is different between the two locations, not completely different but for sure different since Kassala is close to the sea and El-Geneina has a more dry climate. Our model is worked out specifically for El-Geneina which means it will do a really good 1-step prediction for that exact climate while doing a good but not great 1-step prediction on a similar climate. However for the 7-step prediction both our models do almost

exactly equally well when measuring the MSE for El-Geneina and Kassala. However the normalized variance is a lot higher for Kassala indicating that we are way off for more predictions than in El-Geneina. This is a key factor when doing predictions, we do not want estimations to be way off hence we can conclude that our model generally works well for Kassala but has more deviations with really bad predictions than for El-Geneina. The easiest way to explain this is due to climate differences. The reason for the MSE not differing that much between the two for 7-step can also be explained by just saying that making a 7-step predictions is very hard and doing a great one will be hard and conditions must be really good but that is not our case hence we get okay and similar total error for Kassala and El-Geneina.

7 Conclusions

Now to discuss table 5. We can see all our models do a good job at the 1-step prediction yielding a low MSE on the test data except for the Box Jenkins model which seems to struggle a bit more. It is somewhat clear for the 1-step that we can choose any one of the three other models; however we actually chose the polydiv since it is the most simplistic model and we like to keep it simple. For the 7-step prediction neither one of them are great and none of them are really bad. The lowest MSE is also for polydiv without input and a close second being Kalman filter without input however we still think Kalman with input is the best just by looking at it, since it does actually follow the trend of the realization better hence we will choose this model for doing the 7 step prediction of Kassala even though the MSE is slightly higher.

It is interesting that using rain as input does not improve our models at all for the test data but rather makes them worse. One of the reasons for this could be that our model for the rain is horrible . We have no way of knowing how much it actually rained each 10-day period during the whole month so our model for that will not have a positive effect since we are actually just guessing how much it rained so using this as input would in fact not help. Another reason is that rain does not have the same effect on vegetation as we think. If we look at figure 29 below when we plot the rain and the vegetation together, the rain is just divided by a constant for clarity, we can see that more rain does not mean more vegetation this does in no way mean that they are uncorrelated but as we can see clearly in the graph, a lot of rain does not mean a lot of vegetation. Rain just leads to vegetation this might also be a solid explanation to why our models with rain as input do not improve the results. After a lot of rain follows a normal amount of vegetation, after a small amount of rain follows the same amount of vegetation this can be seen clearly in figure 29.

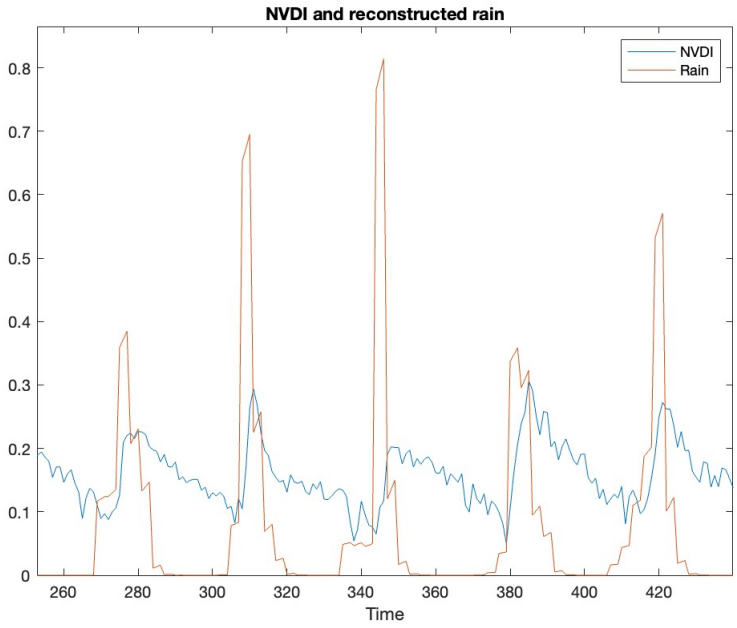
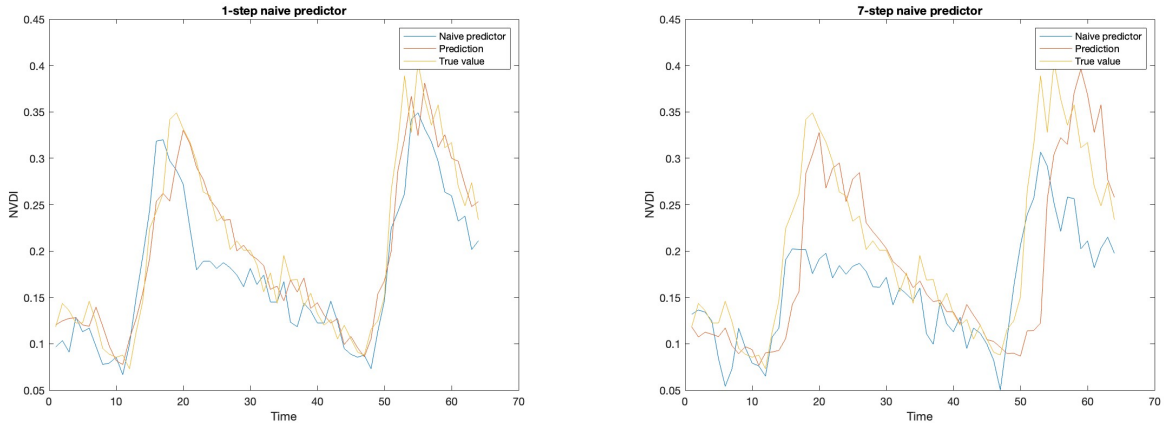


Figure 29: Plot of NVDI against scaled rain data to emphasize argument stated above.

Some concluding thoughts, the reason why polydiv without input works so well is not that hard to understand if looking at the states in the Kalman filter, we can see that our states that does not apply to rain are rather stable and close to our true value as well as them being clearly non-zero meanwhile our rain states are not stable and very close to zero indicating low impact hence indicating rather stationairy data and therefor a timevarying model not helping us that much.Us choosing a low Re for our Kalman filter is due to data being fairly stationary and this worked better. Rw was set to the standard deviation of the residual from our Box Jenkins on validation data as for theory. Also our model working better on the test data than on our validation data can be explained easily enough by just looking at it. Our validation data is much more aggressive than our test data, i.e it has a lot higher variance which is harder for our model to predict, indicating that our test data is more similar to the training data than the validation data is. We thought about using the entire data set when finding a model for the rain for Box Jenkins i.e from 1960 since more data is always better but opted not to because of climate change and using a more recent data seemed more appropriate.

Now to the final boss, is our Naive predictor worse than our best 1-step and 7-step predictor, hopefully our models are better. Our naive predictor was of course that the vegetation next year will be the exact same as this year. To illustrate how they compare we can see figure 30 below with both the naive and the best k-step predictor plotted against the realization of the test data. As we can see in figure 30 of the 1-step our model is much better, but the naive predictor works surprisingly well. The MSE for our naive predictor is two orders of magnitudes larger than our model indicating that our model is not useless but actually good.

Our 7-step predictor also works a lot better than the naive 7-step predictor as can be seen in figure 30 the MSE of the naive predictor is much worse than for our model and the variance of the residual is also greater for the naive predictor.



(a) 1-step naive predictor.

(b) 7-step naive predictor.

Figure 30: Plots of the naive predictor compared to our best models and the true El-Geneina data.

And with that we would like to round off and say thank god we did not do all of this in vain and thank you for very interesting and intriguing lectures as well as a great course.