Time series models for Rain-Vegetation data in Sudan

Mohamed Abrash
Paulina Ibek

December 2023

1 Introduction

In this report, we delve into the application of various time series models for analyzing vegetation index data obtained from two cities in Sudan. Our objective is to assess the performance of different models in capturing the temporal patterns and dynamics inherent in the vegetation index.

The models under consideration encompass an Autoregressive Moving Average (ARMA) process, a Box-Jenkins model incorporating precipitation as an input, and an adaptive Kalman model where the coefficients dynamically vary over time. Our analytical approach involves the construction of k-step linear predictors for each model, with a subsequent evaluation of their predictive accuracy.

To quantify the performance of these models, we employ standard metrics such as the Mean Squared Error (MSE) and the variance of the prediction error. These metrics provide valuable insights into the reliability and precision of our models in forecasting vegetation index values.

This report is a project submission for the Time Series Modelling course at Lund University. Consequently, the nomenclature and notation employed herein strictly adhere to the established conventions of the course. To maintain conciseness, we abstain from extensive exploration of the theoretical underpinnings, as these are thoroughly addressed in the course book [1]. Instead, our emphasis is on interpreting and discussing our selected methodologies, along with presenting the results derived from their application.

2 Reconstruction of rain data

The provided rain measurements from El-Geneina and Kassala were collected on monthly basis (total monthly precipitation) between year 1960 and 2000. Whereas the vegetation index values were measured every 10 days between 1982 and 2000. This discrepancy in the measurement frequency poses a challenge to our analysis and we must resolve it before proceeding . In this section we provide two different methods to address this discrepancy. The first method involves rebinning the data using deterministic trends to ensure temporal alignment. The second method employs stochastic models for rebinning. By applying one of these methods, we obtain precipitation and vegetation index measurements aligned on a 10-day basis. This crucial step simplifies the modeling procedure and enables us to construct a Box-Jenkins model for the vegetation index and rainfall, as demonstrated in the subsequent sections.

First method: deterministic linear interpolation

The first method involves adding two extra points between the monthly measurements by linearly interpolating the data, resulting in three measurements every month. It is important to note that the introduction of two extra points increases the total monthly precipitation. To address this, we normalize the resultant precipitation values by dividing them by 3, under the assumption that, on average, this conserves the measured monthly precipitations.

Worth noting here is that this interpolation approach is somewhat simplistic. The actual rain measurements may not conform to deterministic linear trends on a monthly basis, indicating that our estimation is, in essence, an educated guess.

Second method: Stochastic reconstruction (Kalman filter)

The second method for reconstructing rain data involves treating rain as a stochastic process and applying the Kalman filter. We begin by assuming a simple structure for rain, focusing on short-term behavior (every 10 days). We model 10-days cumulative precipitation measurements X_t using an $AR(1)$ process:

$$X_{t+1} = aX_t + e_t, \quad t \in \mathbb{N} \tag{1}$$

where a is a real constant and e_t is white noise with variance σ_e^2. The cumulative monthly rain is expressed as:

$$Y_k = X_{3k} + X_{3k-1} + X_{3k-2} + \omega_k, \quad k \in \mathbb{N}. \tag{2}$$

Here, measurement errors ω_k are assumed to be white noise.

In state space representation, we regard $S_k = [X_{3k} X_{3k-1} X_{3k-2}]^T$ as the internal state of the "precipitation system," updating as an $AR(1)$ process:

$$S_{k+1} = a^3 \mathbf{1} S_k + \mathbf{e_k} \tag{3}$$

where $\mathbf{1}$ is the identity matrix, and $\mathbf{e_k}$ is the model noise given by Equation 4.

$$\mathbf{e_k} = \begin{bmatrix} e_{3k+3} + a e_{3k+2} + a^2 e_{3k+1} \\ e_{3k+2} + a e_{3k+1} + a^2 e_{3k} \\ e_{3k+1} + a e_{3k} + a^2 e_{3k-1} \end{bmatrix}. \tag{4}$$

Monthly measurements are the sum of the state vector components and can be written as:

$$Y_k = C \cdot S_k + \omega_k. \tag{5}$$

Assuming a and σ_e^2 are known, we reconstruct the states S_k, representing precipitation measured every 10 days, using the Kalman filter. We choose $\mathbf{A} = a^3\mathbf{1}$, $\mathbf{C} = [1\ 1\ 1]$, and the covariance matrix corresponding to Equation 4 is given by:

$$\mathbf{R_e} = \sigma_e^2 \begin{bmatrix} 1 + a + a^2 & a + a^3 & a^2 \\ a + a^3 & 1 + a + a^2 & a + a^3 \\ a^2 & a + a^3 & 1 + a + a^2 \end{bmatrix} \tag{6}$$

For details on Kalman filter construction in a state space representation, refer to [1]. We test our Kalman interpolation method by simulating data from an $AR(1)$ process and reconstructing it. To use the Kalman filter for rain reconstruction, we must choose or estimate the values for a and σ_e^2. This can be done either by direct measurement or by estimating these constants using inference techniques based on the statistics of the monthly rain Y_k. However, in the studied case, the choice of a can be made inconsequential by choosing σ_e^2 and σ_ω^2 wisely. This is done by selecting a arbitrarily in the interval $(-1, 1)$ and picking σ_e^2 to be relatively large compared to σ_ω^2 and a. Essentially, we compensate for our lack of knowledge of a by increasing the state noise, indicating a lack of trust in our specific choice for a. Furthermore, by selecting σ_ω^2 to be small, we control how close the reconstructed monthly total is to the measurements $\mathrm{Var}[Y_k - (X_{3k} + X_{3k-1} + X_{3k-2})] = \mathrm{Var}[\omega_k]$.

Due to the nature of the problem, we cannot directly verify which of the methods is better at reconstructing the 10-days precipitations. However, we can at least require that the total sum of the reconstructed rain matches the total sum obtained from measurements, ensuring that we do not alter the precipitation amount in the long run. Additionally, since our methods operate by rebinning monthly rain measurements, we can also demand a similar requirement on a monthly basis, namely $Y_k = \sum_{j=0}^{2} X_{3k-j}$.

Results

The results of the Kalman interpolation are shown in comparison to the linear interpolation in figures 1 and 2. Table 1 shows the effect of choosing different values for the state and measurement noise.

Table 1: This table presents a comparison between two different methods employed for rebinning rain data from monthly to 10-day resolution. Two measures have been selected to assess these methods. The first is a global measure, ensuring the conservation of the total precipitation sum for the entire dataset. The second measure calculates the total deviations within a month between the reconstructed and measured data. In the table, Y_k denotes the measured monthly precipitation, where k represents the month index. X_t represents the 10-day reconstructed precipitation, with the index t iterating over 10-day periods.

Method	Global deviations	Local deviations
Expression	$\lvert\sum_k Y_k - \sum_t X_t\rvert$	$\sum_k \lvert Y_k - \sum_{j=0}^{2} X_{3k-j}\rvert$
Linear interpolation	2.62×10^{-10}	2.72×10^{3}
Kalman reconstruction $\left(a = 0.8, \sigma_e^2 = 10^2, \sigma_\omega^2 = 0.01\right)$	0.06	0.10
Kalman reconstruction $\left(a = 0.8, \sigma_e^2 = 10^4, \sigma_\omega^2 = 0.01\right)$	6.24×10^{-4}	1.00×10^{-3}
Kalman reconstruction $\left(a = 0.8, \sigma_e^2 = 10^4, \sigma_\omega^2 = 10^{-5}\right)$	1.00×10^{-6}	6.24×10^{-7}

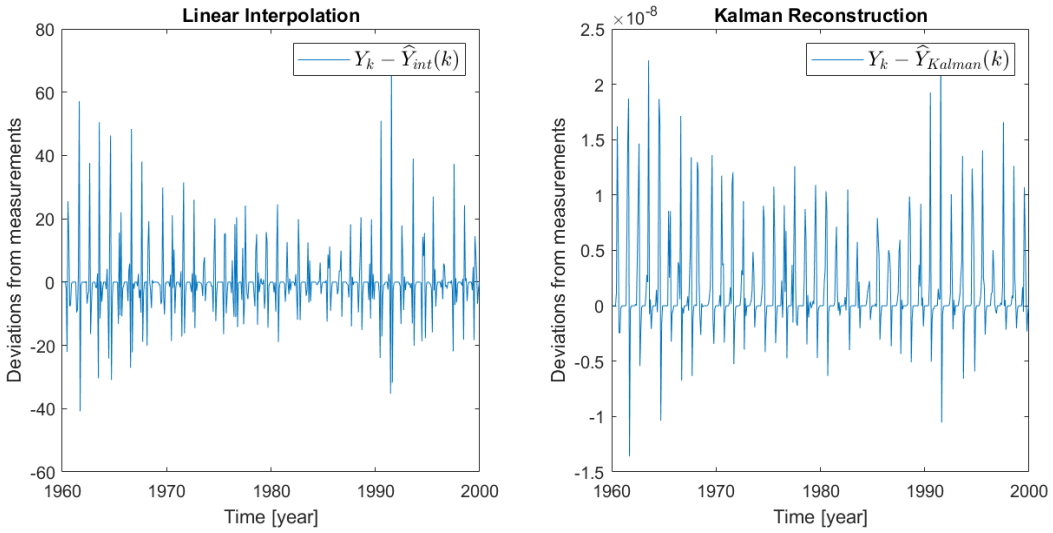

Figure 1: Plots of local deviations between monthly precipitation measurements and the sum of the three constructed precipitation values in the same month. The left side shows errors produced when the data is reconstructed using linear interpolation, while the right side shows errors when data is reconstructed using the Kalman filter. The Kalman-reconstructed data exhibit much better agreement with the measured monthly precipitation. In these figures, $\widehat{Y}(k)$ refers to the sum of the three reconstructed points within month k such that $\widehat{Y}(k) = \sum_{j=0}^{2} X_{3k-j}$.

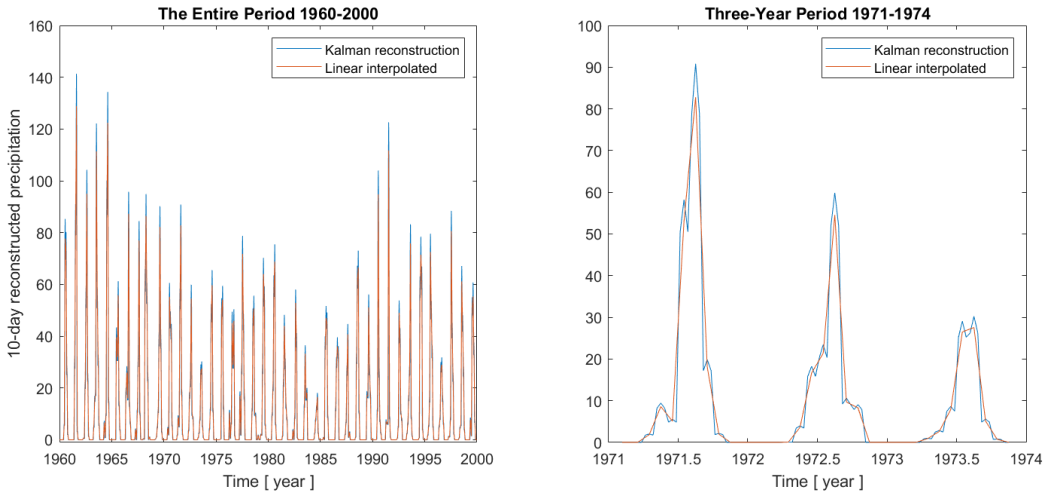

Figure 2: Comparison between reconstructed precipitation values using linear interpolation and the Kalman filter. The figures exhibit similar behavior on large intervals, with expected local disagreements. The linearly interpolated data shows more regular behavior, while the Kalman-reconstructed data is more stochastic.

One artifact of the Kalman reconstruction is that we have not provided any constraint on the values allowed for the state vectors and as a result some of the obtained points in the reconstruction are slightly bellow zero. Of course negative rain is nonsensical, however, has no consequences on our later models since we are focused on modeling the NDVI values. On the other hand, if our models are to be combined with other perhaps dynamical models then using negative values for the rain is indeed a serious problem. In what follows, since these points offer no big contribution to the total precipitation, we simply set them to zero. Alternatively, the magnitude of these points is made vanishingly small by choosing appropriate variances for the Kalman filter as seen in table 1. Finally, we note that the reconstructed precipitation is not normally distributed according to the D'Agostino-Pearson's K2 test. To simplify later development we look for a transformation that can stabilize the variance and result in a normally distributed data. No trivial transformation was found using a Cox-Box test. Nevertheless, by testing various options we found the transformation $(log((x^{0.5}+1))$ to produce a normally distributed data by the D'Agostino-Pearson's K2 test.

3 Model without input for El-Geneina

First a model without precipitation as external input was created for the NDVI. The first step was to rescale the NDVI from the interval [0, 255] to [-1, 1]. In order to asses necessary prepossessing steps, the data itself, as well as the auto correlation function (ACF) and partial auto correlation function (PACF) for the NVDI data were plotted, for number of lags 1/4 the length of the data. Since the data might need a transformation to stabilize the variance, a BC normality plot was created to check for an optimal λ.

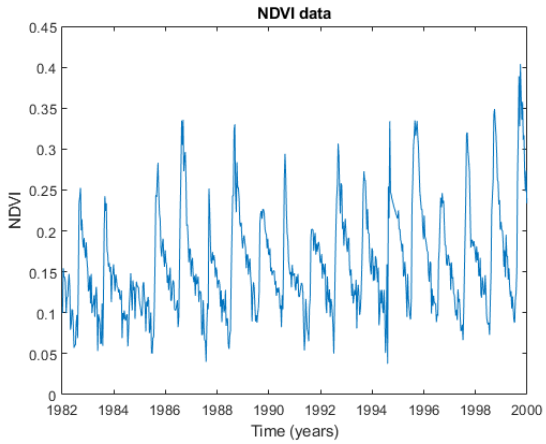

Figure 3: The normalized difference vegetation index time series data.

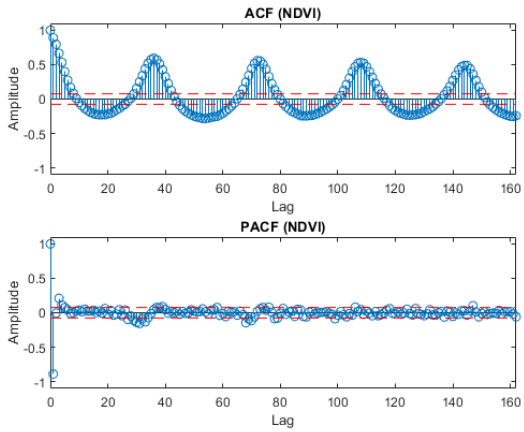

Figure 4: The ACF and PACF plots for the NDVI data.

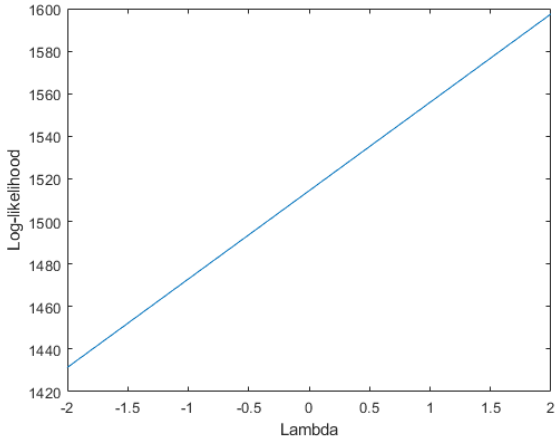

Figure 5: Box-Cox normality plot, which inidcates which transformation is needed to stabilize the variance.

From the time series itself, one can conclude that there are no outliers which need to be taken care of. The ACF plot has a ringing behaviour, which is a strong indicator of seasonality. It also seems to decay slowly, which could mean that there is a trend. The BC norm plot was inconclusive, since there does not seem to be a λ which maximizes the log likelihood, however, we decided to perform some sort of transformation, which was chosen empirically. A couple transformations worked fairly well, and for this particular model we settled for y_t^{-2}. A deterministic linear trend was then estimated using the least squares method, and removed from the data. The ringing in the ACF plot suggested a seasonality of 36, which was removed using appropriate differentiation. By this point the ACF and PACF plots look like this.

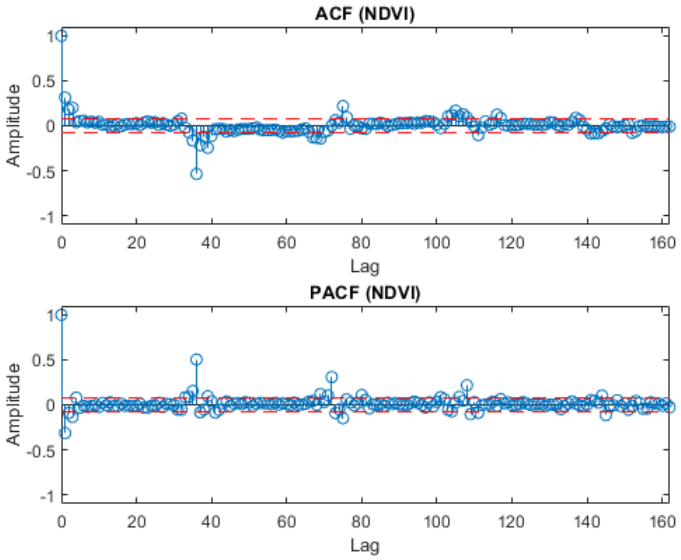

Figure 6: ACF and PACF plots after preprocessing.

Now the preprocessed data is split into training, validation and test sets with a 70/20/10 ratio. The proposed model is an ARMA(2, 2) with the following parameters

$$\mathrm{A(z)} = 1 - 0.4971(\pm 0.112)z^{-2}$$
$$\mathrm{C(z)} = 1 + 0.2983(\pm 0.04401)z^{-1} - 0.3853(\pm 0.1121)z^{-2}.$$

The resulting model residual of the training data together with its ACF and PACF plots as well as normality plots can be seen in the figure below.

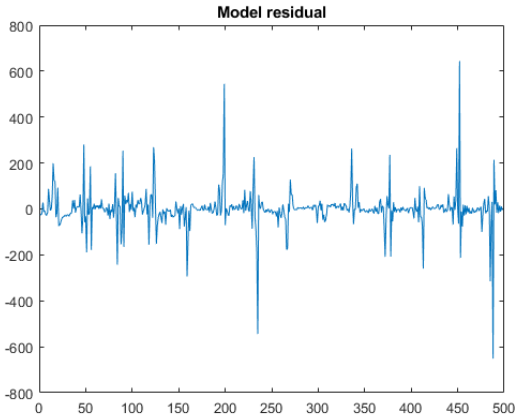

Figure 7: The model residual of the training data.

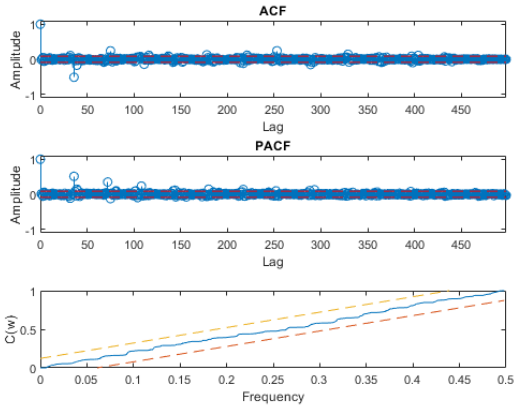

Figure 8: The ACF, PACF and normplot of the model residual of the training data.

According to all metrics, the residual is white. We mainly use the Monti test, which has a value of 6.52 ($\leq$ 36.42) in this case. Said model was also applied on the validation and the test data, both of which generated white model residuals. The Monti test values were 33.06 and 25.93 for the validation and test sets respectively.

Next, a one-step optimal linear prediction was performed for the NDVI using the obtained model in the transformed domain.

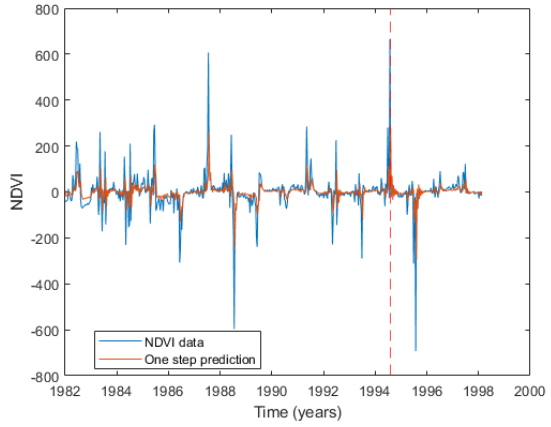

Figure 9: One-step optimal linear predictor for the training and validation data, which are split by the dashed red line.

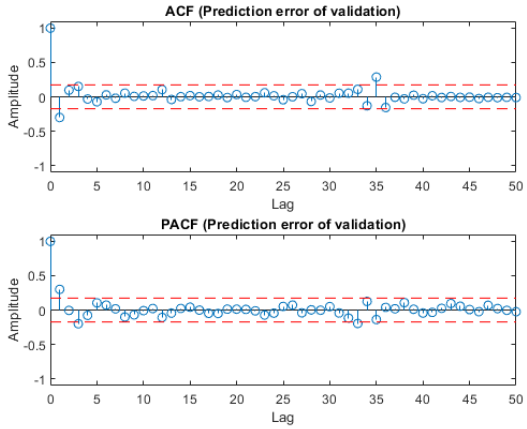

Figure 10: ACF and PACF plots of the prediction error of the validation set.

The error for the one-step predictor for validation data set was white according to all metrics, and had a Monti test value of 26.70. The variance of the error 6172.9. which is high, however the domain is also scaled differently. The variance of the prediction residuals divided by the variance of the validation set was 0.7. The naive predictor in our case is that the NDVI is the same each year, which implies a shift in the data by 36 measurement steps. In the figure below is the naive predictor in the "processed" domain.

The naive predictor had a variance of 18535, which is about three times higher than that of the one-step predictor. The variance of the prediction error for the naive predictor divided by the variance of the data was 2.1, so three times higher than for the one-step predictor. Finally, we did the same procedure for the test data, where the error of the prediction was again white, with a Monti value of 26.23. The normalized variance of the prediction residual for the test data was 1.1549, which is again lower than that of the naive predictor.

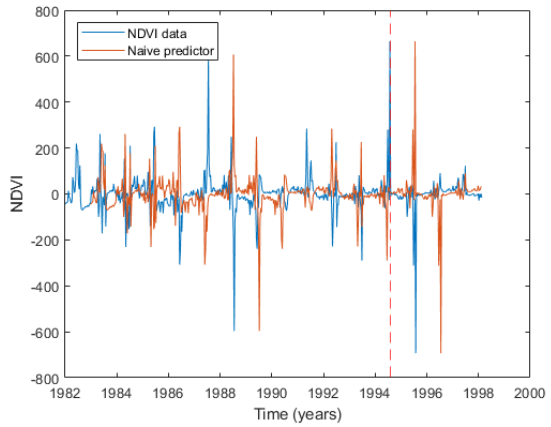

Figure 11: Naive predictor in the processed domain

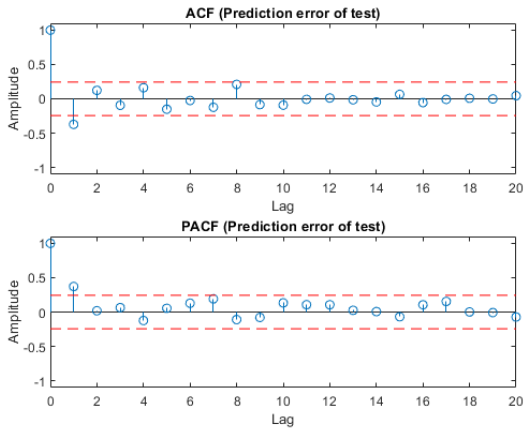

Figure 12: ACF and PACF for the test data for the one step predictor

Next the 7-step prediction was conducted in the same way as before.

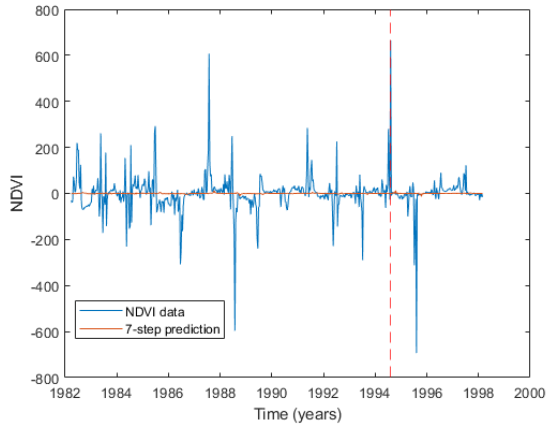

Figure 13: The 7-step prediction for training and validation data, where the dashed red line markes he beginning of the validation data.

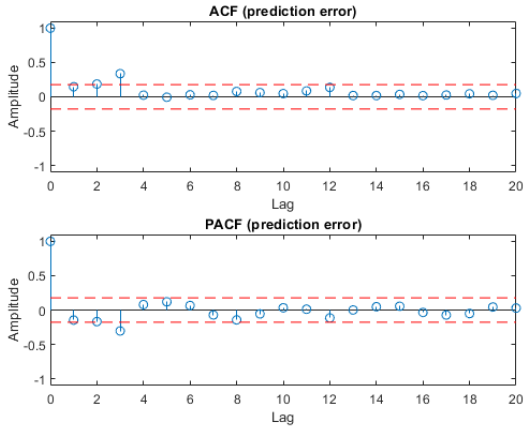

Figure 14: The ACF and PACF plots for the 7-step validation data

From the ACF plot, it seems that the error can be modeled like an MA(3). According to theory, for a k-step prediction, the error should be modeled as an MA(k-1). This tells that the model we have probably could be improved. However, one would expect a 7-step predictor to perform worse than a one-step predictor. The variance of the error was 5707.4 and the variance of the error divided by the variance of the validation data was 0.6472. Finally the 7-step prediction was made on the test data.

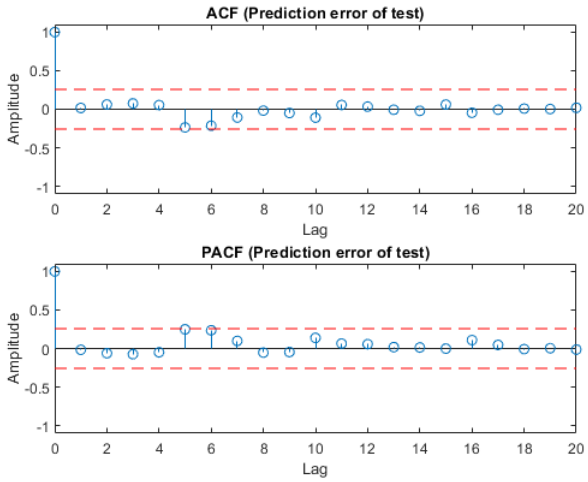

Figure 15: ACF and PACF of the test data for the 7-step prediction error

From these figure it almost looks like the error could be modeled like white noise. However, we see that if the confidence intervals were a bit more narrow, it could look like an MA(6) process. For this to happen, the confidence intervals had to be adjusted to a significance level of 0.2. The variance of the prediction error for the test data was 722.99, and by normalizing it against the variance of the test data it became 0.9994. Finally, we transformed back the data and the prediction to the original domain. Once again we compared the one-step and 7-step prediction to the naive predictor.

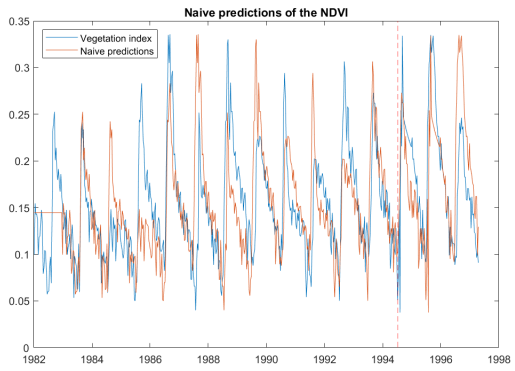

Figure 16: The NDVI data together with the naive predictor in the original domain.

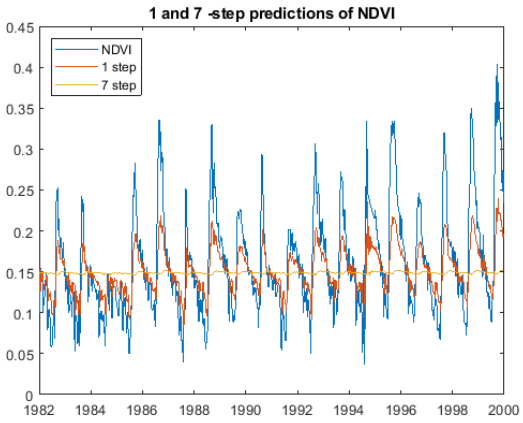

Figure 17: The NDVI data together with the one-step and 7-step optimal linear filter predictions.

From the figure above, it seems that the naive predictor does a better job than in the processed domain. In order to quantify which prediction works the best, we calculated the mean squared error between the different predictors and the NDVI data. The results were 0.0020 for one stop prediction, 0.0044 for 7-step and 0.0045 for the naive predictor, which means that our one-step predictor was the most accurate followed by the 7-step and naive predictor, which performed virtually the same. From this we can conclude that the model for NDVI without external input that we constructed in this case describes the data well, and can be used for predictions.

4 ARMA model for the precipitation

An ARMA model for the precipitation data is build in this section as it can assist us in the construction of the Box-Jenkins model that we intend to consider for the vegetation index. The ARMA model will also allow us to make a k-step linear predictor for future precipitation values which is then used for the prediction of future vegetation-index values.
We begin our modeling of the precipitation by inspecting the ACF and PACF of the data seen in figure 19. Removing the apparent 36 periodicity by including a z^{-36} component to the AR part of the model, results in the ACF and PACF functions seen in figure 18 which further suggests MA coefficients at lag 1 and 2 among others. Preceding this process until we obtain Gaussian residual gives the following ARMA model:

$$A_x(z) = 1 - 0.417(+/-0.04052)z^{-3} + 0.2063(+/-0.02995)z^{-12} - 0.4637(+/-0.04012)z^{-36}$$
$$C_x(z) = 1 + 1.021(+/-0.02423)z^{-1} + 0.9553(+/-0.02764)z^{-2} + 0.0419(+/-0.02432)z^{-12}$$

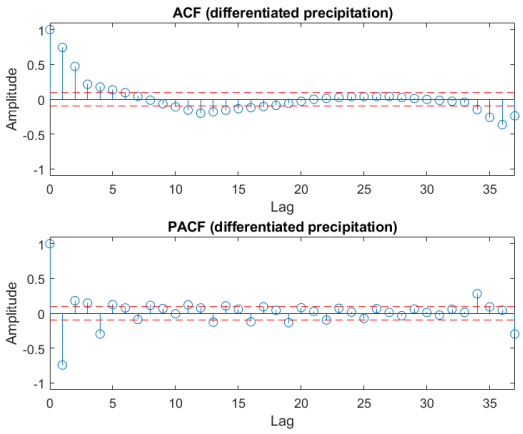

Figure 18: ACF and PACF plots of the model residual resulting from modeling the precipitation with one AR coefficient at lag 36. The plot suggests MA coefficients at lag 1 and 2 and AR coefficient at lag 1.

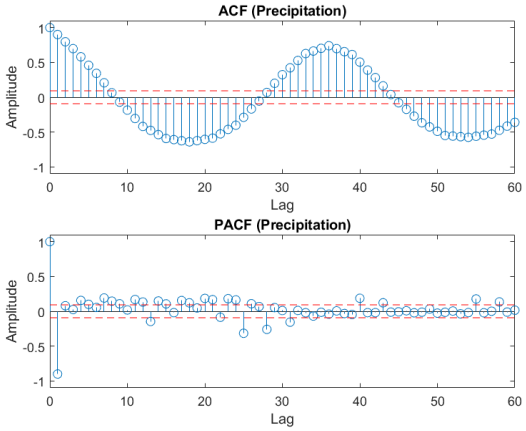

Figure 19: ACF and PACF plots of the precipitation training data. The ACF shows a ringing behaviour which indicates a seasonality of 36.

This model was found using PEM and it results in white noise residual according to the Monte test with test value $25.5 < 36.4$. A plot showing the ACF and PACF of the model residual with 95% confidence intervals is shown in figure 21 as well as a normality plot which shows that our residual is indeed normally distributed and validates the confidence intervals at use.

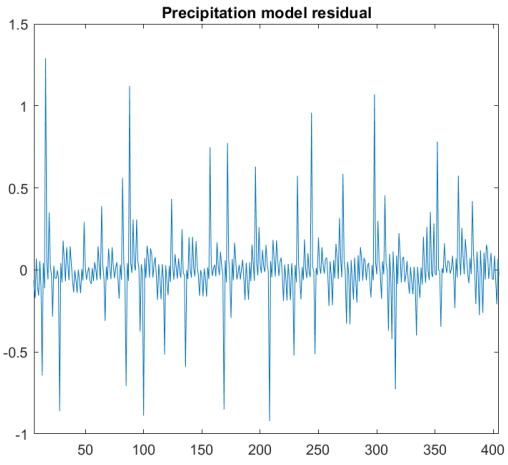

Figure 20: Plot of the model residual.

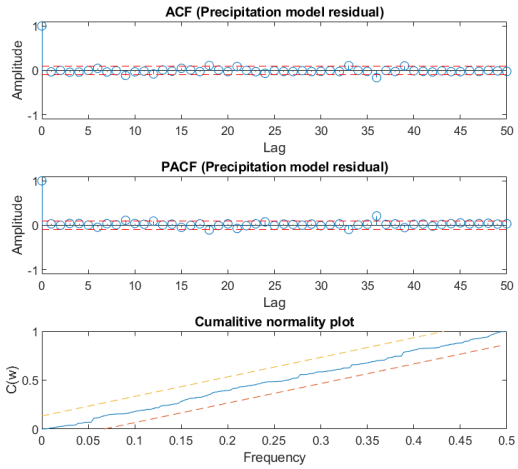

Figure 21: ACF and PACF plots for the precipitation model residual as well as a normality plot of the residual. The plots suggest that the residual can be considered as white noise. The cumulative normality plot shows that the residual is within the 95% confidence intervals of a Gausian distribution.

5 Box-Jenkins model

Here we construct a Box-Jenkins model for the NDVI where we consider the precipitation as an input. Figure 24 shows the cross correlation function between the input (precipitation) and the output (NDVI) which indicates that indeed there is cross corelation which may be used in the modeling of the vegetation. Before we start modeling, certain pretreatment must be performed on the NDVI data similar to what we did in the previous section on (Model without input for Elgeneina), however, for the rest of the project we simplify this treatment to, scaling the data to be in the interval (-1,1), subtracting the mean and taking the log in order to obtain normaly distributed data (which we prefer in order to carry out the modeling and make use of confidence intervals). These transformations reversed at the end of the modeling when we present k-step predictions. The normality of the data ,after taking the log, is verified by using the D'Agostino-Pearson's K2 test. A normality plot is shown in figure 22.

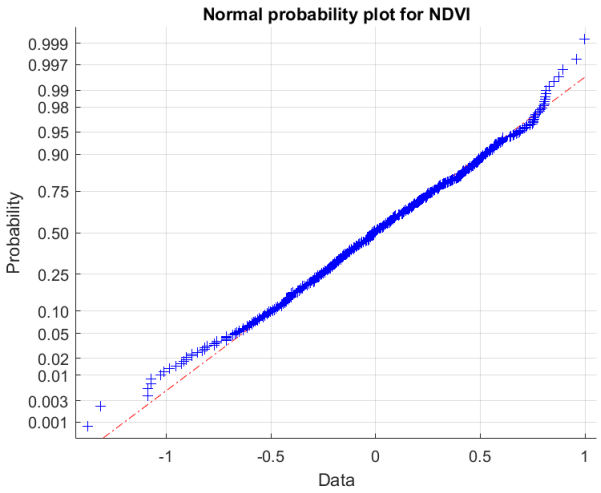

Figure 22: Normal probability plot for the log of the rescaled NDVI data

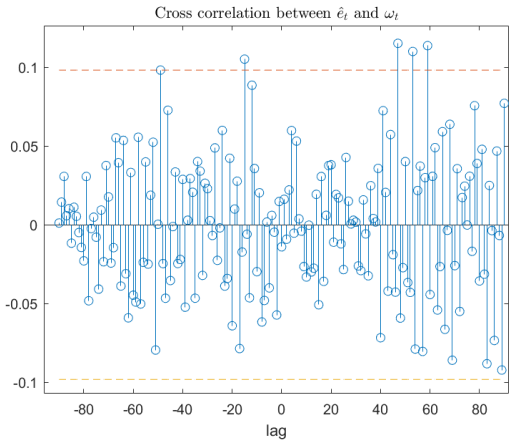

Figure 23: Cross correlation function between the prewhitened input $\frac{C_x}{A_x}y_t$ and the innovation noise of the input ω_t.

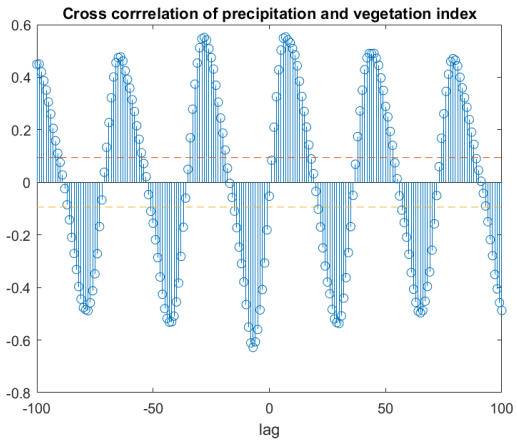

Figure 24: The cross correlation function between the input (precipitation) and the output (vegetation). It shows that there is strong correlations periodically repeating at various lags which indicates that the rain can be used as an input for the modeling of the vegetation index.

A Box-Jenkins model takes the form

$$y_t = \frac{Bz^{-d}}{A_2}x_t + \frac{C_1}{A_1}e_t \tag{7}$$

with y_t and x_t being the output and the input respectively while e_t is a zero mean white noise process. The ARMA model found for the input can be used here to estimate orders of the polynomials B,A_2 and the delay d, following the treatment in [1]. The cross correlation function between the pre-whitened input $\hat{e} = \frac{C_x}{A_x}y_t$ and the input's innovation white noise ω_t is seen in figure 23 The figure shows no cross correlation at lags less that 45. However, intuetivly we suspect that the precipitation at a certain period has an effect on the vegetation in the near future and we therefore suspect that the delay factor d in the Box-Jenkins model should be less than 6 (which is equivalent to two months). This can also be seen in the precipitation-vegetation cross correlation in figure 24 . The reason this is not seen in the cross correlation plot seen in figure 23 might be attributed to the influence of the model noise term ($\frac{C_1}{A_1}e_t$) which is assumed to be negligible in this analysis. It is worth noting that the cross correlation plot seen in 23 is dependent on the particular model we use for the precipitation.

Guided by these arguments we empirically test out various candidates of transfer function structure $\frac{Bz^{-d}}{A_2}$ and we identify a suitable model structure

$$Bz^{-d} = b_0 z^{-4} \tag{8}$$

$$A_2 = 1 + a_1 z^{-1}. \tag{9}$$

Using this found structure and finding the best fitted coefficients using PEM we form a residual as

$$\tilde{e_t} = y_t - \frac{Bz^{-d}}{A_2}x_t. \tag{10}$$

The cross correlation between this residual and the precipitation , as seen in figure 26, shows that for the chosen structed we manage to subtract/ explain much of the initial correlation seen in figure 24. Next we proceed to model the ARMA part of the Box-Jenkins model. That is, we try to find polynomials A_1 and C_1 such that

$$\tilde{e} = \frac{C_1}{A_1}\nu_t \tag{11}$$

where ν_t is a white noise process. Following the same procedure we used before to model a process as an ARMA, we begin by plotting the ACF and PACF of $\tilde{e}$, figure 25. The ACF shows a ringing behaviour while the PACF shows only one significant coefficient at 1. This indicates an AR model of order 1

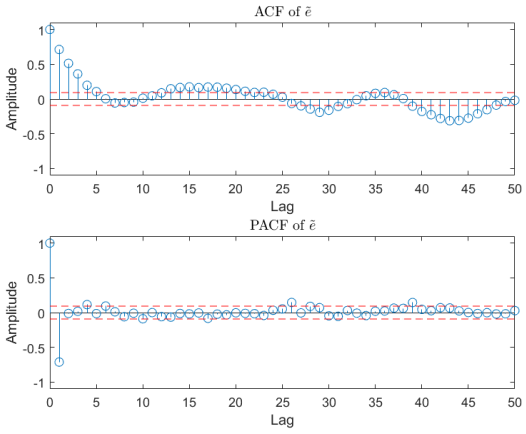

Figure 25: ACF and PACF of $\tilde{e}$. The ACF shows an oscillatory behaviour while the PACF part indicates an AR coefficient of lag 1.

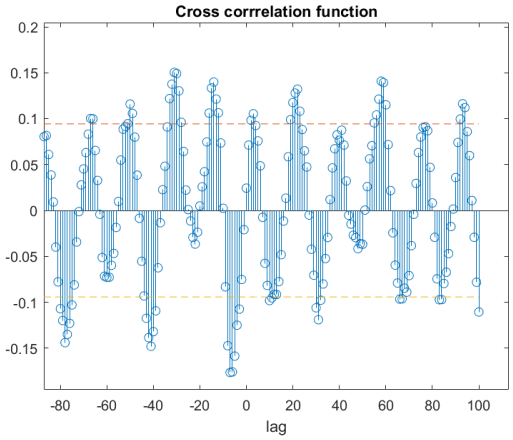

Figure 26: Cross correlations between $y_t - \frac{Bz^{-d}}{A_2}x_t$ and x_t

Using PEM we find an optimal AR(1) model :

$$A_1 = 1 - 0.7147(+/- 0.03479)z^{-1} \tag{12}$$

which results in a residual which is considered white by the Monte test. Combining the polynomial structures we found for A_1,A_2,C_1,B and d we find the appropriate coefficients using PEM and test if the resultant Box-Jenkins model give a white noise residual. The found model is

$$B(z)z^{-4} = 0.06992(+/- 0.009336)z^{-4} \tag{13}$$

$$A_1(z) = 1 - 0.7148(+/- 0.03489)z^{-}1 \tag{14}$$

$$A_2(z) = 1 - 0.8994(+/- 0.02556)z^{-1} \tag{15}$$

The model residual is found to be white according to the Monte test (20.6). Figures 27 and 28 show the model residual and its ACF and PACF.

Initially, more sophisticated models were tested, including coefficients at lags (35-37) to account for the annual periodicity, however, when using PEM to identify these coefficients, they offered little improvement to the model and removing them did not have any effect on the performance.

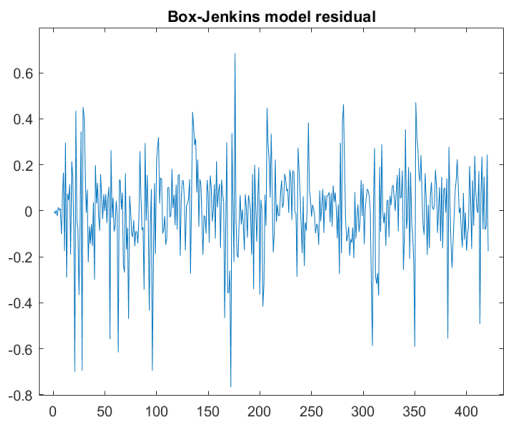

Figure 27: Box-Jenkins model residual

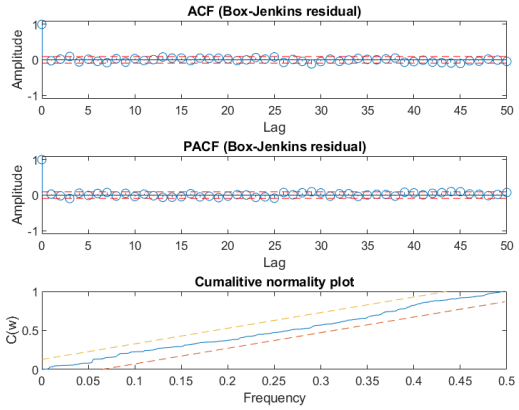

Figure 28: ACF and PACF of the Box-Jenkins model residual and its normality plot.

5.1 k-step predictions with the Box-Jenkins model

Next, we test our models ability to replicate the data by constructing the optimal k-step future predictions. We first rewrite the Box-Jenkins model in the form of an ARMAX.

$$K_A y_t = K_B x_t + K_C e_t \tag{16}$$
$$K_A := A_1(z)A_2(z) \tag{17}$$
$$K_B := A_1 B(z) z^{-d} \tag{18}$$
$$K_C := A_2 C_1 \tag{19}$$

A k-step predictor is the expectation value of the vegetation index at time $t+k$ given data up to time t. Following the construction in [1], we can decompose the ARMAX model into terms that depend on data up to time t and terms that depend on future data as follows:

$$\hat{y}_{t+k} = \hat{F}\hat{x}_t + \frac{\hat{G}}{K_C} x_t + \frac{G}{K_C} y_t \tag{20}$$
$$\hat{x}_{t+k} = \frac{G_x}{C_x} x_t \tag{21}$$

where $\hat{F}, \hat{G}$ and G_x are defined by the Diophantine equations:

$$\frac{K_C}{K_A} = F + z^{-k}\frac{G}{K_A} \tag{22}$$
$$\frac{FK_B}{K_c} = \hat{F} + z^{-k}\frac{\hat{G}}{K_C} \tag{23}$$
$$\frac{C_x}{A_x} = F_x + z^{-k}\frac{G_x}{A_x} \tag{24}$$

Note that, generally, a future prediction of the vegetation-index involves the future predictions of the precipitation which we also need to update recursively. Figure 29 shows the 1- step and the 7-step predictions both for the precipitation and for the NDVI (vegetation-index). For the precipitation we see that the predictor is able to foresee future values without much deviations even for 7-steps ahead. This is due the regular structure of the rain data with the strong periodicity. As for the vegetation things are abit different. The one step predictions are able to predict future values while the 7 step predictor is only able to capture the

overall periodicity of the vegetation index. Compared to the predictions seen for the ARMA model **??**, we see that the Box-Jenkins model offers a big improvement to the 1-step and 7-step predictions. This is not surprising as the precipitation values have a strong influence on the vegetation and including it as a part of the model does ,as we have seen, improve the predictions.

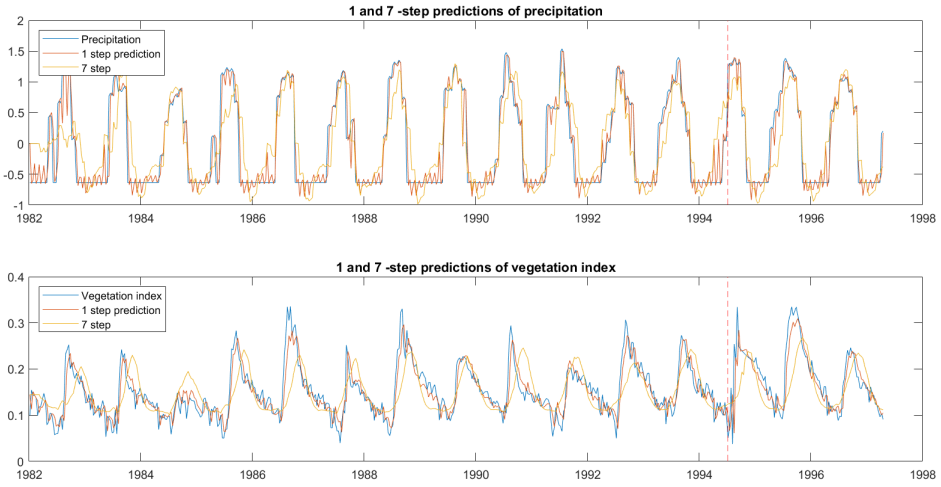

Figure 29: 1- and 7-step predictions of the precipitation and vegetation data from Elgeneina using the models developed in this project.

The variance of the 1-step and 7-step prediction errors on the validation set are 0.27 and 0.63 where we normalised by the variance of the data. We also calculate the MSE (mean square error) of the prediction errors on the validation set and we get 0.057 and 0.145 for 1-step and 7-step predictions respectively. As discussed above these metrics indicate much improvement on the ARMA model we constructed without the use of precipitation data. This is also reflected visually in figure 29. Finally, we plot the ACF of the prediction errors as seen in figure 30. Theoretically, the ACF of a k-step prediction should show the character of an MA(k-1). This is indeed the case for the 1 step prediction errors. However, the 7-step prediction errors has a ringing behaviour and significant coefficients beyond a lag of 6. This missmatch between observations and theory can be attributed to factors such as finiteness of the data (theoretically the prediction errors should be MA(k-1) in the limit when we have large number of data points). Another explanation might be the imperfection of our model. Although, we have modeled the data as a Box-Jenkins we recognise that the data is the result of the complex and chaotic earth weather system which most certainly does not generally follow a Box-Jenkins model.

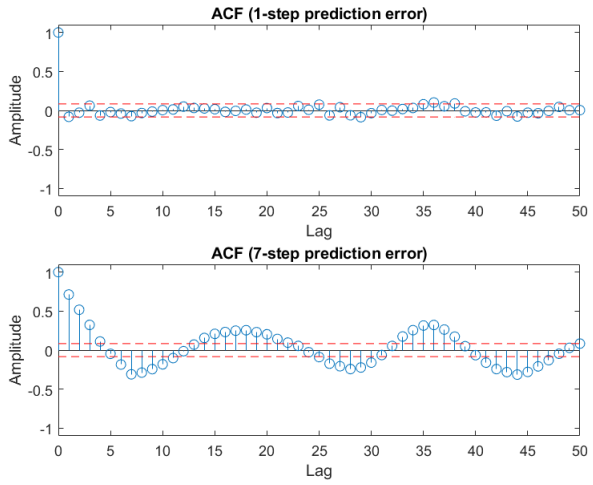

Figure 30: The ACF of the 1- and 7-step prediction errors obtained for the validation set using the found Box-Jenkins model.

6 Time varying model

In this section we attempt to improve our Box-Jenkins model by allowing the model coefficients to vary over time, allowing us to capture any changes to the model. To do this we use the Kalman filter which we will set up in this section. We start by writing the ARMAX process in state space representation:

$$y_t = C_t X_t + w_t \tag{25}$$
$$X_{t+1} = X_t + \mathbf{e}_t \tag{26}$$

where X_t is a vector containing the estimates of the model's coefficients at time t while C_t is a row vector containing data up to time t. For the model given above we get

$$C_t = [-y_{t-1} \quad -y_{t-2} \quad +x_{t-4} \quad +x_{t-5} \quad +e_{t-1}] \tag{27}$$
$$X_t = [K_A^1(t) \;\; K_A^2(t) \;\; K_B^4(t) \;\; K_B^5(t) \;\; K_C^1(t)]^T \tag{28}$$

where $K_A^i(t)$ refers to the estimate of the ith coefficient of the polynomial K_A at time t. This formalism will allow us to set up the Kalman filter according to the formulation in [1]. Furthermore, since the Kalman filter gives the one step prediction we can construct a k-step prediction by recursively predicting one step at a time, meanwhile replacing future values of x_{t+r} and e_{t+r} by their expectation values. For the white noise this value is zero while for the input x we can use the predictions we constructed above in the section on the Box-Jenkins model. That is to predict y at time $t+k$ we write:

$$\widehat{y}_{t+k} = \widehat{C}_{t+k}\widehat{X}_{t+k} \tag{29}$$
$$\widehat{X}_{t+k} = X_t \tag{30}$$
$$\widehat{C}_{t+k} = [-\hat{y}_{t+k-1} \quad -\hat{y}_{t+k-2} \quad +\hat{x}_{t+k-4} \quad +\hat{x}_{t+k-5} \quad 0\,] \tag{31}$$
$$\tag{32}$$

1- and 7-step predictions of future vegetation index values are shown in figure 32 in the original domain. Here we used the coefficients obtained for the Box-Jenkins model as our initial state vector values, as for the variance of the measurement noise we used the estimate obtained from the static Box-Jenkins residual ($\approx 10^{-2}$) while the variances of the state vector components were tuned by visual inspection. The evolution

of the state vector components (the estimated coefficients) are shown in figure 31. We can see that , apart from the jump after 1994 the coefficients are relatively stationary. We further investigate tuning the variance of these coefficients to allow them to change and catch local variations but that only resulted in reductions of the prediction quality. For ease of comparison we also plot the Box-Jenkins (non-dynamic model) predictions along side the Kalman predictions in figure 32.

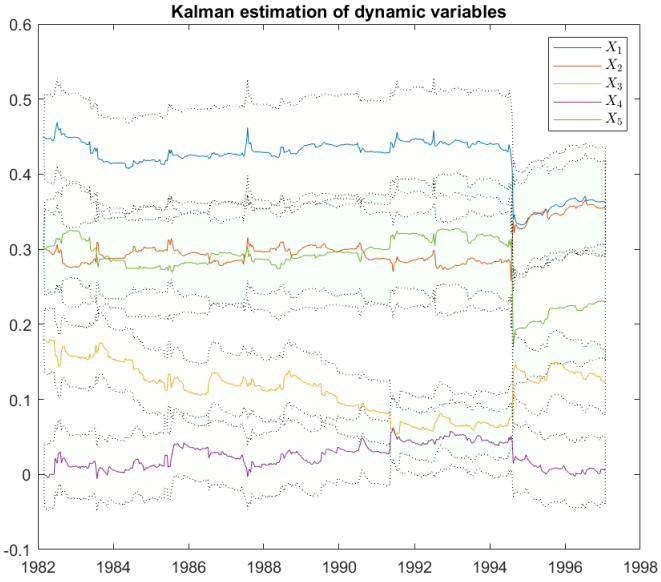

Figure 31: The evolution of the model coefficients as calculated by the recursive Kalman filter.

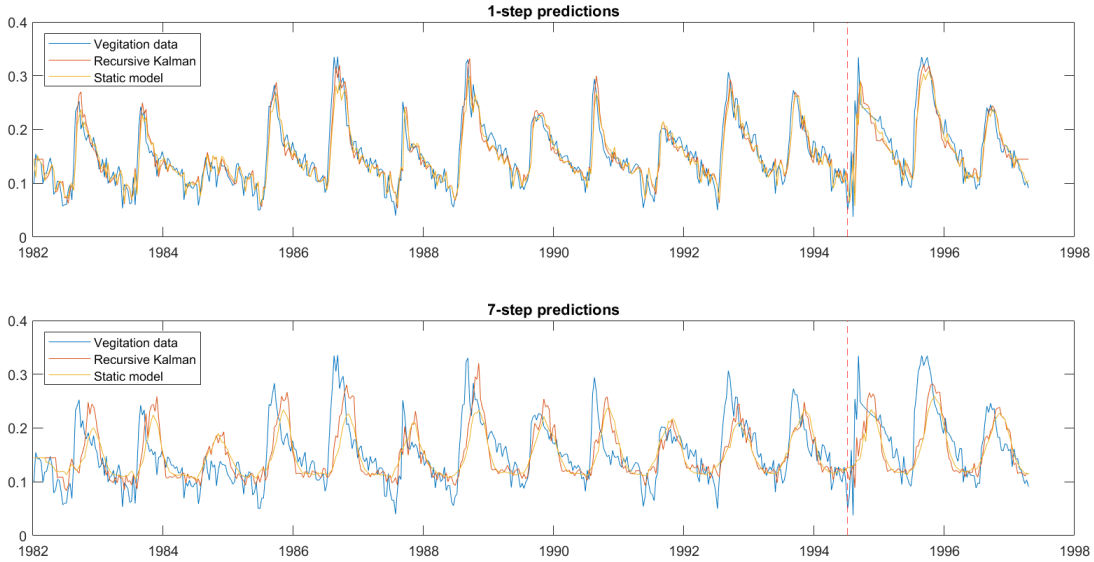

Figure 32: 1- and 7-step predictions of future vegetation-index data using the Kalman filter and the Box-Jenkins model.

It is clear, from these plots that the two models perform similarly with the stationary Box-Jenkins model being slightly better. To confirm this we calculate the normalized variance of the prediction error on the validation set using the Kalman filter and we find 0.26 and 0.72 for the 1- and 7-step prediction errors. Figure 34 summarizes our findings from all the different models we considered.

When we compare the quality of the predictions and the variances of the predictions errors we see that the adpative Kalman model is not able to outperform the static Box-Jenkins model. Our interpretation for this is the data might in fact be stationary which means that there is little gain from using an adaptive model. To confirm this we perform a stationarity test such as the Kwiatkowski–Phillips–Schmidt–Shin (KPSS). When applied to the combined training and validation sets the KPSS test confirmed that the data is stationary. Another reason for why the adaptive model is incapable of outperforming the static model could be the fact that our chosen model structure is too simple and that through our modeling procedure we stripped the model from any coefficients that capture the local variations of the data and only kept the coefficients that captured the 'global' overall behaviour of the data. It is possible that without these 'local' coefficients, which we deemed insignificant, the adaptive model simply does not have the facility to compensate for these missing factors.

We next investigate if we can reduce our adaptive model without loosing performance. To do so we start removing the least significant coefficients from the model and investigate the prediction quality. It was found that the model can be reduced down to to components without much of a loss when it comes to prediction quality. The model structure then becomes:

$$y_t + ay_{t-1} = bz^{-4}x_t + e_t \tag{33}$$

The normalised variance on the validation set using this reduced adaptive model was calculated to be 0.31 and 0.63 for 1-step and 7-step prediction errors respectively. The predictions are shown in figure 33

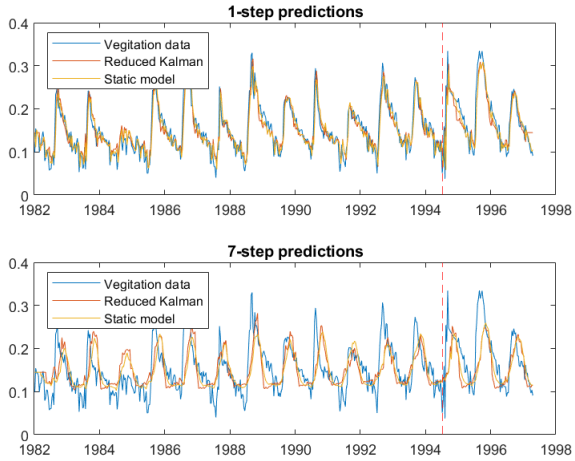

Figure 33: 1- and 7-step predictions using the reduced Kalman model.

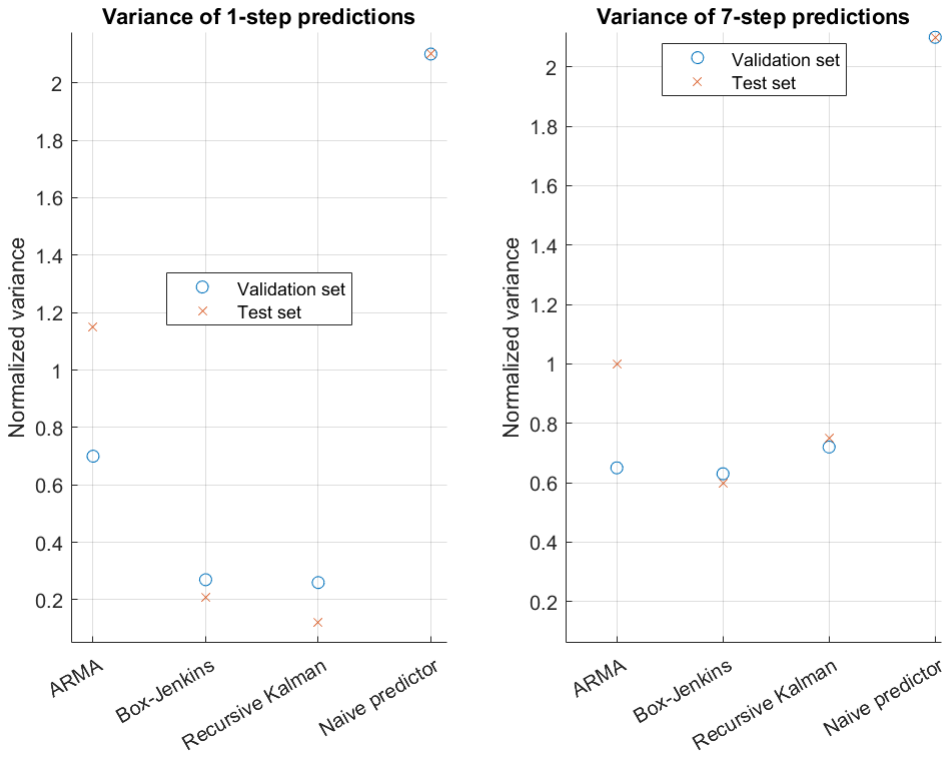

Figure 34: Model comparison using the normalized variance of the prediction errors as a metric. These variances are calculated both for the validation data in 'o' and for the test data in 'x' both for 1-step prediction errors (left figure) and for 7-step prediction errors (right figure)

Figure 34 offers a comparison between the main models we considered in the modlling of the vegetation index. It show the normalised variance of the prediction errors for 1-step and 7-step predictions both on validation and test data. Generally for statistical models, the performance on the validation set is higher

that on the test set since the models have been tempered and tuned on the validation set. This is indeed what we see for the ARMA model. However, for the Box-Jenins and the recursive Kalman we see no such drop in performance.

7 Modelling for Kassala

Now we want to test the adaptive Box-Jenkins model on the Kassala site. The first step was to reconstruct the rain data using Kalman filtering in the same way as for El Geneina. Afterwards we use the same Box-Jenkins model that we got for El-Geneina location, and utilize the Kalman filter to vary the coefficients. We also did a version where the Box-Jenkins coefficients were re estimated for the Kassala location. Given the polynomial model

$$y(t) = \frac{B(z)}{F(z)}u(t) + \frac{1}{D(z)}e(t)$$

the parameters were

$$\begin{aligned} \text{B(z)} &= 0.004936(\pm 0.005731)z^{-4} \\ \text{D(z)} &= 1 - 0.6806(\pm 0.03512)z^{-1} - 0.1955(\pm 0.03387)z^{-36} \\ \text{F(z)} &= 1 + 0.9926(\pm 0.002989)z^{-1}. \end{aligned}$$

For this part of the project, we used all of the data at once, and did not divide into training, validation, and test sets, since we're only interested in testing the model, rather than finding it.
From the one step predictions we see that the model works better for El Geneina than for Kassala. Even after adjusting the BJ model, there are still peaks in the predictions which overshoot the actual data. The mean squared error, together with the normalized variances for the different cases are displayed in the table below.

	El Geneina	Kassala	Kassala adjusted BJ model
Recursive Kalman MSE	0.0012	0.0014	0.0014
Recursive Kalman Normalized var	0.0074	0.0106	0.0106
Box-Jenkins MSE	0.00080	0.0011	0.0011
Box-Jenkins normalized var	0.0049	0.0080	0.0083

Table 2: One step predictions in the different cases

From the errors we see that the model works better for El Geneina than for Kassala. The differences in results between the two Kassala predictions are very small, if any, so adjusteing the Box Jenkins model didn't seem to make a large difference.

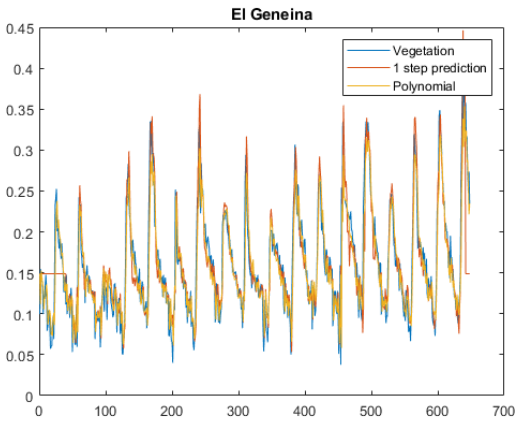

Figure 35: One step prediction for static model and Kalman filtering for El Geneina location.

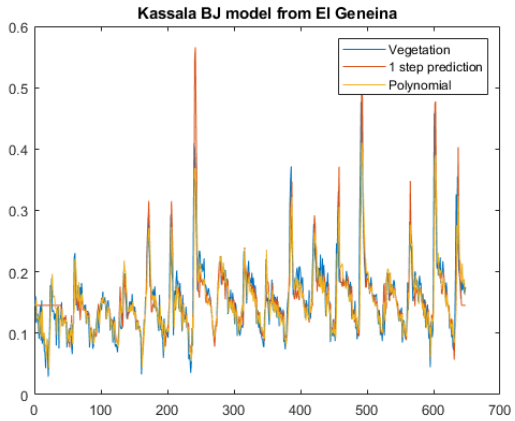

Figure 36: One step prediction for static model and Kalman filterinf for Kassala location with BJ model from El Geneina.

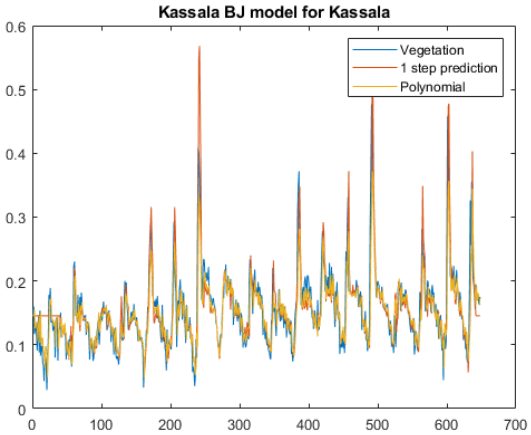

Figure 37: One step prediction for static model and Kalman filterinf for Kassala location with BJ model from Kassala.

For the seven step prediction, again the model for El Geneina seems to work better. Once again there is one peak for both Kassala results, where the predictions are much higher than the actual vegetation data.

	El Geneina	Kassala	Kassala adjusted BJ model
MSE Kalman	0.0037	0.0040	0.0040
norm var Kalman	0.0026	0.0303	0.0305
MSE polydiv	0.0026	0.0030	0.0028
norm var polydiv	0.0155	0.0224	0.0204

Table 3: 7-step predictions in the different cases

Observing the MSE and normalized variance in the 7-step case, we once again see that the error as well as its variance is smaller for El Geneina. For the adaptive prediction, both BJ models seem to work equally well for Kassala, with the normalized variance being only slightly worse in the adjusted Box Jenkins version.

For the polynomial division prediction, the version with the adjusted BJ model works better, since it has lower error and error variance compared to the Kassala prediction with the old BJ model.

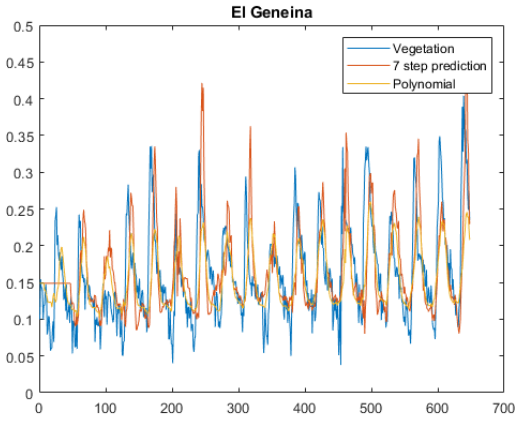

Figure 38: Seven step prediction for static model and Kalman filtering for El Geneina location.

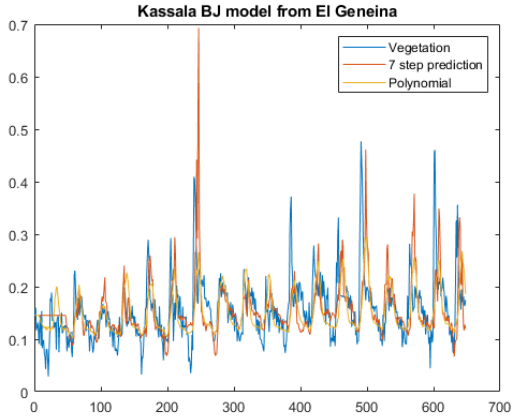

Figure 39: Seven step prediction for static model and Kalman filterinf for Kassala location with BJ model from El Geneina.

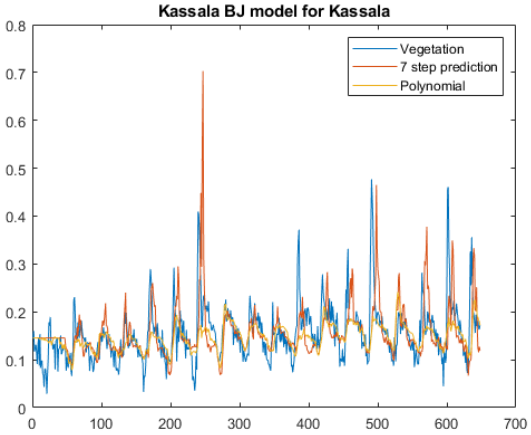

Figure 40: Seven step prediction for static model and Kalman filtering for Kassala location with BJ model from Kassala.

This model worked fairly well for both locations, but any differences can of course be due to the fact that El Geneina and Kassala are about 1500 km apart. Although, by just looking at the map, it looks like they would have similar biomes, maybe the vegetation reacts a bit differently to the precipitation, or maybe the precipitation patterns differ.

References

[1] A.Jakobsson. *An introduction to time series modeling*. 4th ed. Additional notes. Malmö, Sweden: Studentlitteratur, 2021. ISBN: ISBN.