Predicting the Temperature in Tåstrup

a time series analysis project

Project report
FMSN45: Time series analysis

We have not collaborated with anybody, other than the teaching staff in the course.

Victoria Lagerstedt & Kajsa Hansson Willis
Group 8
Fall 2024

1. *Introduction and theory*

Trying to predict the weather has been a human preoccupation for as long as mankind has existed. It is a fundamental cornerstone of agriculture and one of the most heated debates regarding sustainability. One can therefore make the point that the accuracy of climate predictions is important. As Robert Frost poetically wrote:

Some say the world will end in fire,
Some say in ice.
From what I've tasted of desire
I hold with those who favor fire.
But if it had to perish twice,
I think I know enough of hate
To say that for destruction ice
Is also great
And would suffice.
~ Robert Frost [Poetry Foundation n.d.]

Inspired by the poem, this report makes an effort to try to create models that can predict the temperature patterns in the Danish city of Tåstrup based on past measurements of environmental data. These models are offered as both stand-alone predictors and with the entire code and process presented to give both a practical tool and a deeper understanding of the modeling process.

Three different models will be presented; each one adding complexity to the previous. In part A of the project, discussed in 2.1, a simple ARMA predictor is presented. This model uses only previous measurements of the air temperature to form its prediction. In part B, discussed in 2.2, an ARMAX model is presented. This model uses input from data of net radiation in the area to create the prediction as well. In the third part of the project, part C discussed in 2.3, the model in part B creates the foundation for a Kalman filter that allows for updating parameter estimates as time passes. This is done for both estimating the future external input and the future output.

These models are then compared and analyzed in Chapter 3. The project is concluded in Chapter 4.

The theories used while producing the temperature predictor are the ARMA model, ARMAX model, Box-Jenkins and the Kalman filter. They each have their own strengths and can complement each other well. The ARMA(X) models are suitable for structured predictions with timeinvarying coefficients. This is a contrast from the Kalman filters favor with dynamic forecasting and added flexibility.

The Box-Jenkins methodology provides an appreciated structure for developing the ARMAX model. This model allows for input and is useful when causal relationships are identified or suspected between factors. However, when there is no relationship, the added input is misleading and may cause deficiencies in the model. For instance, a few different environmental data sets were provided together with the temperature data, but only the net radiation was used. The data sets were compared to each other and they all had a similar pattern. It was hypothesized that there was another reason that they looked quite similar and that added inputs probably would not add much value to the model, since the net radiation looked similar and could cover those aspects already. Instead, it might just add more complexity and insignificant parameters. Therefore it is important to use the ARMAX model only for processes where it is likely that there is a strong relationship between the input and output to ensure that the added complexity leads to improved accuracy. Therefore, predictions from the ARMAX model are expected to be better than predictions from simpler models, at least if there is a causal relationship between the input and output.

The ARMA model is simpler and will suffice in many situations. However, the model is thus in general expected to perform worse than more complicated models. It can be considered better for short-term predictions as well because of its simplicity.

The Kalman filter is the most complex of the theories used, and it is expected to add further accuracy because of its dynamical property. However, it adds computational requirements and if the model is stationary there will be no added benefits. Thus, this method is not worth using in these cases. This also means that the Kalman filter is expected to work well on longer lags. If a model that can use information that is unknown today is useful, the Kalman filter is the easiest implementation of that.

2. *Process*

In this stage of the project, the focus was on creating three robust models to predict future temperatures in the area. The strategy was to follow the modeling procedure methodically, which can be observed in Figure 1. This procedure starts with examining the data provided, estimating the model order and parameters and then checking if it works through examining if the residual is white noise. If the residual is not white noise more information can probably be extracted from the data and used to form a better prediction. However, there is value in minimizing the amount of parameters. Therefore, considerable consideration has to be taken when balancing the order of the model. As the data was the same throughout the process, the first steps only had to be done for the air temperature data in part A and the net radiation data in part B. Whenever there were difficulties in the code, they would primarily be approached by simulating data to be able to debug.

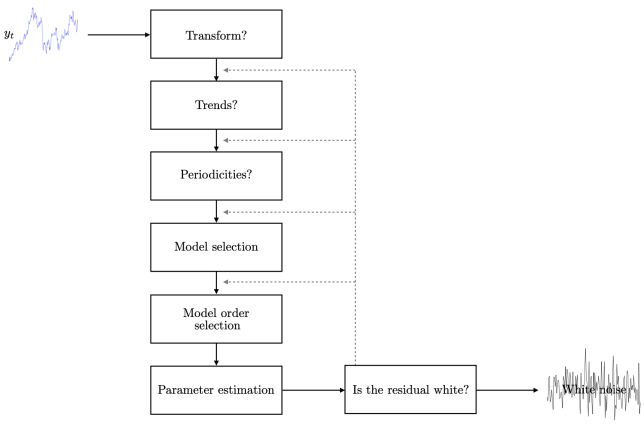

Figure 1: The modeling procedure in time series analysis [Jakobsson 2024]

In the modeling procedure, Equations 1 and 2 will be used to determine the polynomials for an ARMA process so they are presented here.

$$C(z)\,\hat{y}_{t+k|t} = G_k(z)y_t \qquad \text{Equation 1}$$

$$C(z) = A(z)F_k(z) + z^{-k}G_k(z) \qquad \text{Equation 2}$$

2.1 *Part A*

In the first part, part A, the task was to form a predictor of the air temperature using only previous measurements of air temperature. The instructions given were to use a 6–10 week period as modeling

data, a few weeks following that as validation data, and to use the week directly after as well as an arbitrary week during another season as the two test data sets.

2.1.1 Creating the Modeling Data and Examining it Initially

The first thing done was thus to determine which weeks of the air temperature data should be used as modeling data. This was essentially done by isolating the temperature data and plotting. A clear season was observed in the full data set, and the first period that did not look too chaotic and seemed to follow the season was selected. A length of 8 weeks seemed reasonable, which corresponds to 1 344 hours.

First, the data was evaluated. A Box-Cox normality plot was done to test if there was a need for transformation in the data. This was done with the entire data set. This gave a lambda value of 0.9495, which is very close to one and it was thus determined that there was no need for a transformation. A test of whether the data followed a normal distribution was also conducted with the entire data set to learn the conditions of the data set since this is an assumption in many theories so this could affect how conclusions can be drawn. As the data set is limited, it was determined that the D'Agostino test would be the most appropriate. The D'Agostino test showed that the data set does not follow the normal distribution, which is important to keep in mind during the handling of the data. The normal distribution test was also conducted on the ACF and PACF of the data set. It was found that the ACF followed the normal distribution whereas the PACF did not. To examine if there were any outliers in the modeling data, a check of the minimum and maximum value of the data set was conducted. The plot of the data was also examined to see if anything looked out of the ordinary, but there were no signs of extreme outliers. Both of the values and the plot were reasonable so handling of outliers was not deemed necessary.

As part of the modeling procedure, the Matlab *testMean* function was used to determine if there were any trends in the modeling data. The function returned a one which indicates that the null hypothesis that there is not a trend should be discarded. However, in the interest of creating as simple a model as possible combined with observing the ACF and the PACF, it was decided that the trend might not be worth incorporating into the model. Instead, the approach was that if it proved difficult to create a good model, one could go back and add a trend later.

The function *plotPlotsWillisLagerstedt*, henceforth called only *plotPlots*, has been developed to plot a function, its autocorrelation function, its partial autocorrelation function, and its norm plot all in one figure. This function was used to examine the modeling data set in order to determine where to begin with the model and the results can be viewed in Figure 2. From the autocorrelation function, the ACF, a season of about 24 is apparent. With this information, it was determined that a differentiation with `[1 zeros(1,23) -1]` should be done. An ARMA process on the differentiated process was estimated by adding one parameter at a time and looking at the ACF and PACF after each addition. However, the ACF and PACF still showed signs that the seasonality was not fully gone. Even when adding 23, 24, and 25 parameters the resulting ARMA process didn't give white noise. Because of this the data was observed even closer and potential trends were the focus. However, it was decided to try to estimate an ARMA on the undifferentiated data with no trend handling instead.

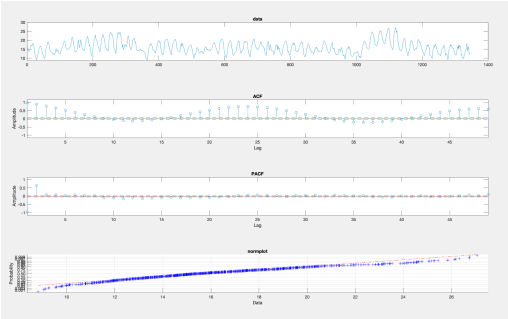

Figure 2: PlotPlots of the modeling data set of the air temperature with the plot, the ACF, the PACF and the norm plot

2.1.2 EstimateARMA Approach

This was done using the *estimateARMA* function to estimate the coefficients of an ARMA model and trying to determine the model order using the ACF and the PACF plots. The function uses the prediction error method for estimation. Through trial and error, by adding coefficients when white noise was not acquired from the residual according to the Monti test and removing insignificant coefficients, an ARMA model with an A(z) with the order `[1 1 1 0 1 1]` and a C(z) with the order `[1 1 1 1 1 0 0 1 1]` generated white noise for lag one and all the coefficients were significant or if they were removed the model would not generate white noise anymore. The Monti test was chosen as a white noise indicator both because of its practicality as the Matlab *checkIfWhite* function could be used easily and because it was deemed robust enough to handle these calculations. This test was used consistently. It was viewed as a strength that it is based on the PACF. These polynomials were:

$$A(z) = 1 - 2.953\,(\pm 0.08302)\,z^{-1} + 2.631\,(\pm 0.1603)z^{-2} - 1.041\,(\pm 0.1601)\,z^{-4} + \\ + 0.3623\,(\pm 0.08282)\,z^{-5}$$

$$C(z) = 1 - 1.556\,(\pm 0.08968)\,z^{-1} - 0.04917\,(\pm 0.08146)\,z^{-2} + 0.8044\,(\pm 0.1203)\,z^{-3} - \\ - 0.1043\,(\pm 0.08919)\,z^{-4} - 0.01965\,(\pm 0.05811)\,z^{-7} - 0.0466\,(\pm 0.04175)\,z^{-8}$$

The *plotPlots* figure with the ACF and PACF can be viewed in Figure 3. The variance of the residual from this model was 0.3656. The normalized variance is 0.0337.

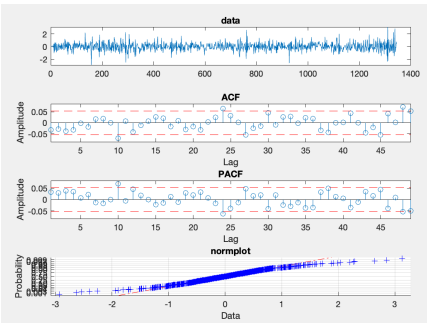

Figure 3: PlotPlots of the residual of the one-step predictor for the modeling data

2.1.3 Checking if the ACF and PACF are Normally Distributed

The next step in the process is to determine if the autocorrelation function and the partial autocorrelation function follow a normal distribution. This was done by filtering the modeling data

through the A- and C-polynomials, adjusting the length, and then using the *plotACFnPACF* function in Matlab. The results were that the D'Agostino-Pearson test indicates that both the ACF and the PACF follow the normal distribution.

2.1.4 Predicting the Future Values

After that, it was time to start with the predictions. This was done on the validation data, which was the two weeks, or 336 hours, directly following the modeling data period. As the values are stable over time polynomial division could be used to construct the equations. The G-polynomial was formed through polynomial division with the A- and C-polynomials, and then the validation data was filtered through the G- and C-polynomials to create predictions of the future y-values. The residuals were then constructed by taking the difference between the true y-value and the predicted y-value according to Equation 3. When the residuals were determined, a check of whether they were white noise or not and an ACF and PACF control was conducted. The model was adjusted until it provided good results for both the modeling data and the validation data.

$$e_t = y_t - \widehat{y_t} \qquad \text{Equation 3}$$

The prediction from the model was then compared to the naive predictor. It was determined that it was appropriate to use the naive predictor that predicted the temperature one time step in the future as the current value, but for longer lags that was deemed an unfair comparison, so for time lags larger than one the temperature 24 hours before was used for the naive prediction.

The naive predictor and the model prediction on the validation data are compared in Figure 4 through 6. For comparison, the variance of the validation data set is 7.60. For a one-step prediction, the variance of the model prediction residual was 0.46 and the variance of the naive prediction residual was 1.42. For the three-step prediction, the variance of the model prediction residual was 1.90 and the variance of the naive prediction residual was 5.30. For the eighteen-step prediction, the variance of the model prediction residual was 4.37 and the variance of the naive prediction residual was 5.30. The normalized variances for the model were 0.06, 0.25, and 0.58 respectively. For the naive predictor, they were 0.19, 0.70, and 0.70 respectively. The *plotPlots* of the residuals, with the residual plot, ACF, PACF, and norm plot, are presented in Figures 7–9 for the one-, three- and eighteen-step predictors, respectively. This information shows that the model that has been created is superior to the naive predictor. For a lag of one, the residual was deemed white noise according to the Monti test. This shows that the model works well for other data sets than the modeling data as well.

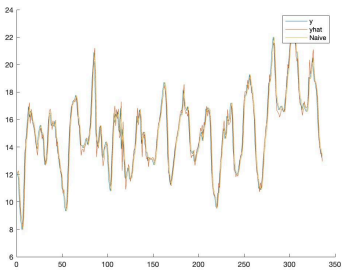

Figure 4: Comparison of the model prediction and the naive predictor for a one-step prediction for the validation data.

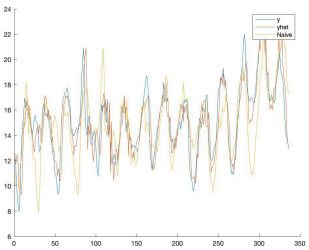

Figure 5: Comparison of the model prediction and the naive predictor for a three-step prediction for the validation data

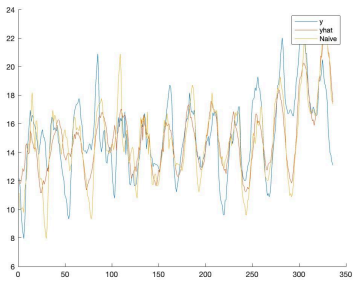

Figure 6: Comparison of the model prediction and the naive predictor for a eighteen-step prediction for the validation data.

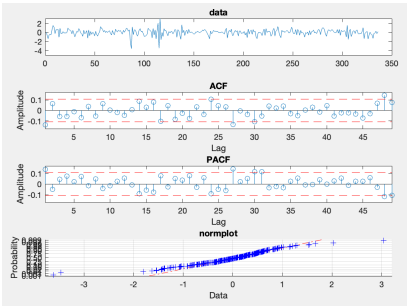

Figure 7: PlotPlots of the residual of the one-step predictor for the validation data.

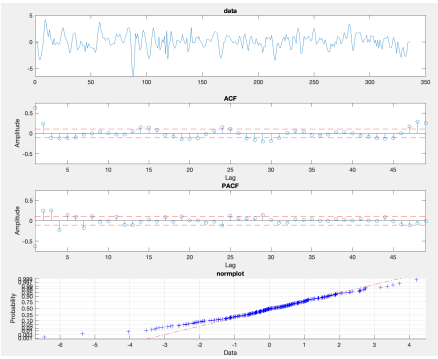

Figure 8: PlotPlots of the residual of the three-step predictor for the validation data.

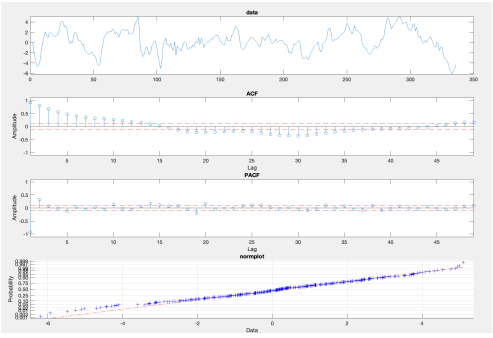

Figure 9: PlotPlots of the residual of the eighteen-step predictor for the validation data.

2.1.5 Testing the model with the test data sets

After the model had been successfully tested with the validation data, it was time to test the model with the test data as well. Two data sets were selected. The first data set was the week, or 168 hours, directly following the validation data. The second data was randomly selected from the data set with the single condition that it should come from a different season of the year.

This was done in the same way as for the validation data. The prediction error was then calculated and plotted with the *plotPlots* function and a Monti test was done. This was done for several values of k-step predictions. The predictions and prediction variance according to the naive predictor were also determined. This was done for both of the sets of test data.

Again, the true values were plotted against the naive predictor and the model's prediction. These plots can be seen in Figures 10 through 15. The variance of the model residual for the first test data set was 0.42 for the one-step prediction, 2.77 for the three-step prediction, and 5.66 for the eighteen-step prediction. The residual variance from the naive predictor for the first test data set was 1.68 for the one-step prediction, 4.66 for the three-step prediction, and 4.66 for the eighteen-step prediction. The variance of the model residual for the second test data set was 0.30 for the one-step prediction, 1.94 for the three-step prediction, and 13.99 for the eighteen-step prediction. The variance of the residual from the naive predictor for the second test data set was 0.32 for the one-step prediction, 18.22 for the three-step prediction, and 18.22 for the eighteen-step prediction. All of the normalized variances are

pictured in Table 1. For the one-step predictions, the residuals from the first test data do not pass the Monti test, whereas in the second test data, they do. However, when observing Figure 10 and examining the variance, the model makes a fair prediction. Bearing in mind that the ACF is not normally distributed and that the model works exceptionally well for the second data set, it is decided that the model is good enough.

Table 1: The normalized variances of various predictions for various k-step predictions and test data sets

Scenario:	*k=1 for the first test data*	*k=3 for the first test data*	*k=18 for the first test data*	*k=1 for the second test data*	*k=3 for the second test data*	*k=18 for the second test data*
Normalized variance from the model:	0.02	0.14	0.28	0.05	0.32	2.27
Normalized variance from naive predictor:	0.08	0.23	0.23	0.05	2.96	2.96

For reference, the variance of the first test data set was 20.10. The variance for the second test data set was 6.15. This means that the variance of the residuals was significantly less than for the data set, which is a requirement for a good model. The variance is lower for the model than for the naive predictor as well, which means that using the model is an improvement from just the naive predictor. One can see in the plots, see Figures 10 through 15, that the model prediction is very close to the true data. This information shows that the model developed for this data set is a better predictor of at least the test data sets than the naive predictor is. As it does a reasonable job of predicting both the test data sets, one can conclude that the model is very reasonable. The variance is considerably smaller than for the test data sets as expected of a reasonable model.

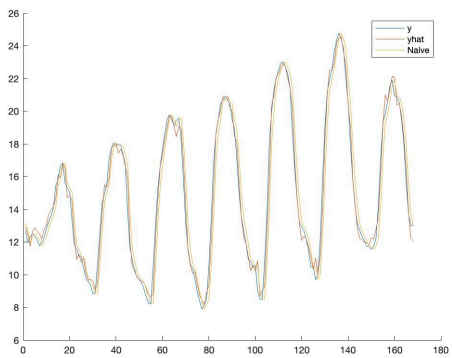

Figure 10: The model one-step prediction compared to the test data and the naive predictor for the first test data set

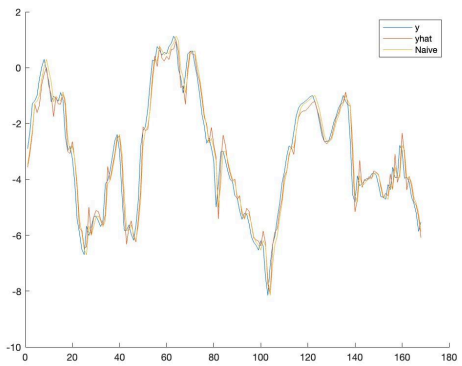

Figure 11: The model one-step prediction compared to the test data and the naive predictor for the second test data set

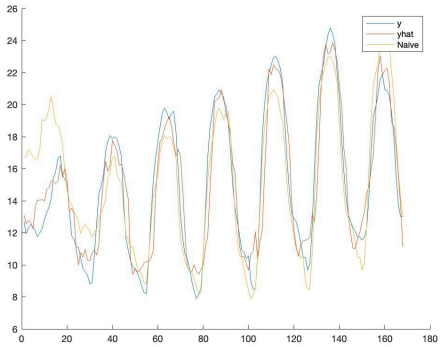

Figure 12: The model three-step prediction compared to the test data and the naive predictor for the first test data set

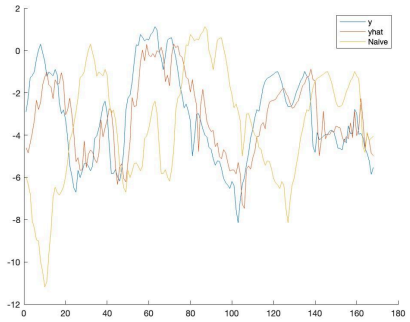

Figure 13: The model three-step prediction compared to the test data and the naive predictor for the second test data set.

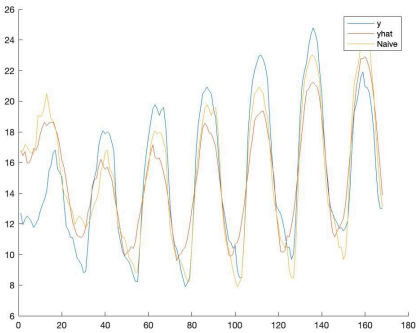

Figure 14: The model eighteen-step prediction compared to the test data and the naive predictor for the first test data set

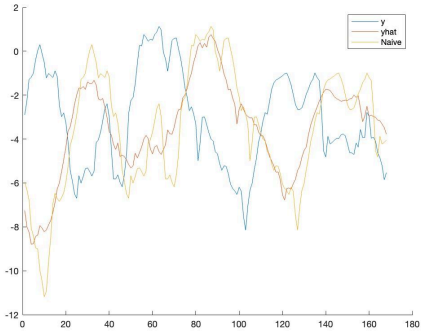

Figure 15: The model eighteen-step prediction compared to the test data and the naive predictor for the second test data set

The ACFs for the prediction residuals can be observed in Figure 16 through 21.

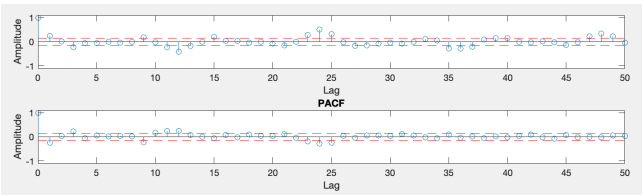

Figure 16: The autocorrelation function for the model one-step prediction for the first test data set.

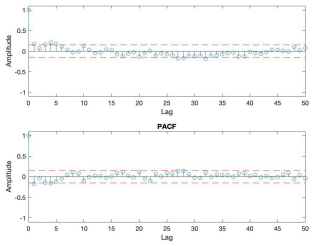

Figure 17:
The autocorrelation function for the model one-step prediction for the second test data set.

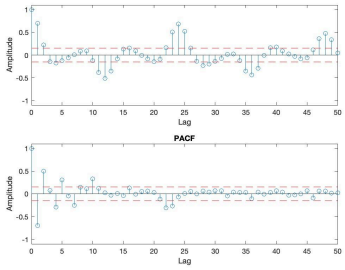

Figure 18: The autocorrelation function for the model three-step prediction for the first test data set

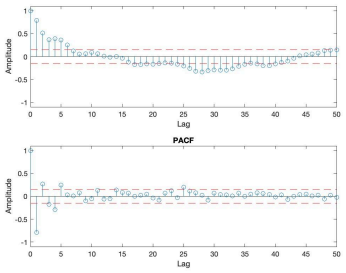

Figure 19: The autocorrelation function for the model three-step prediction for the second test data set.

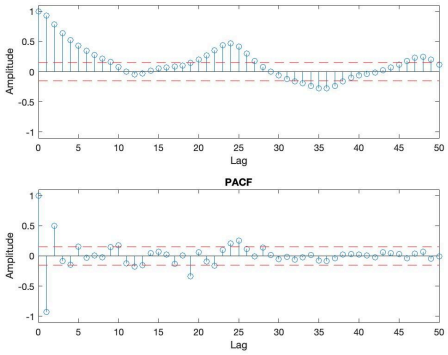

Figure 20: The autocorrelation function for the model eighteen-step prediction for the first test data set

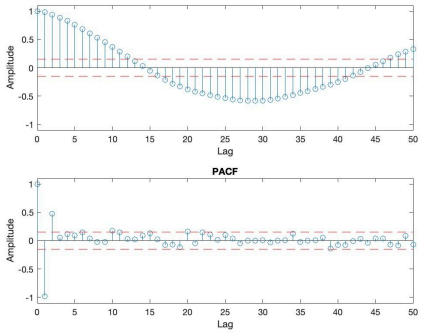

Figure 21: The autocorrelation function for the model eighteen-step prediction for the second test data set

Out of curiosity, the authors wanted to test the robustness of the model and checked other weeklong periods with the model as well. The results of these tests were overwhelmingly that the residual passed the Monti white noise test.

2.1.6 Stand-alone Predictor

When a solid model had been built, the stand-alone predictor was created. It can be found in the file *predCodeA_grp008*. It was built with the A- and C-polynomials that had been previously determined. Predictions were made in the same way as earlier and only the predictions for the indices that were supposed to be tested are returned in the function. The stand-alone predictor was tested with the *examCode* file and called upon in the code for part A.

2.1.7 The model

In general, the result of the modeling procedure has proven to be quite robust. The results for most of the tests were considered white noise, except in specific cases where the results were deemed reasonable anyway. This is especially true when taking into consideration that the data does not follow a normal distribution, and thus, the results cannot be taken at face value. This means that it will be extremely difficult to create a perfect prediction.

As a reasonable model was obtained, it was deemed unnecessary to go back and try some of the things that might have worked well for the data set, for instance adding a trend to the model.

2.2 Part B

In Part B of this project, the purpose was to create a new model to describe the air temperature in Tåstrup. This new model should allow for the use of additional measurements as input into the model, by building a Box-Jenkins model, see Equation 4. A few different data sets were given but additional data sets were allowed.

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

2.2.1 Choosing input data and extracting datasets

The first step was to decide which input data to use. This was done by plotting all of the given input data sets next to each other and next to the temperature data. It was decided that no other external input than the ones given in the course would be evaluated since they were found to be sufficient. After carefully looking at each of the input datasets, the net radiation input was chosen, see Figure 22. The reason for this was that it followed the peaks in the temperature data well, had clear peaks, and didn't seem to have any distinct outliers. The other data sets had some distinct outliers and did not have as high peaks that followed the peaks in the temperature as the net radiation data.

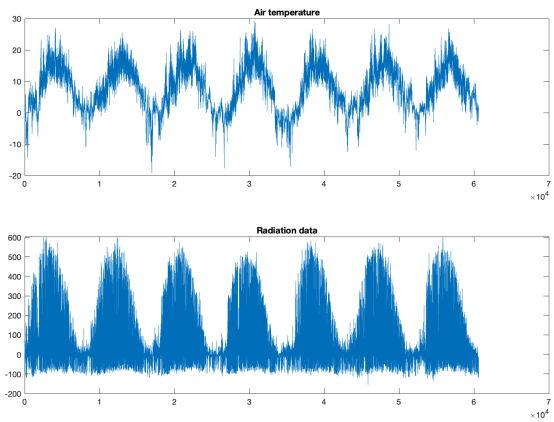

Figure 22: The full data sets of the air temperature and the net radiation

When input data had been selected the modelling data was extracted from both the input and output datasets. The same part of the datasets that were used in part A were used in this part, 8 weeks starting at 3 000 hours in. The validation data and test datasets used in part A were also extracted in this part, in both the input and the output data.

2.2.2 Examining the input data

The input model data was examined using different tests. First a D'Agostino-Pearson's K2 test, using the function *checkIfNormal*, was computed to check if the input model data was normally distributed. A D'Agostino-Pearson test was chosen as a normality check since the data set was limited. The test concluded that the data wasn't normally distributed, which is important to remember when analyzing further. The ACF and PACF of the data was also checked with a D'Agostino-Pearson K2 test. The test concluded that the ACF and PACF for the whole data set were not normally distributed. This means the confidence intervals in the whiteness tests used in this part can't be trusted fully and this was taken into account when running these tests.

A Box-Cox test on the whole input data was used to check whether the data should be transformed. The maximum log-likelihood value occurred when lambda was -0.3 which might suggest a $y^{-0.5}$

transformation of the data. It was decided against doing a transformation of the data and to continue without it. But this was regarded throughout the process and if future issues arose a transformation of the input data would have been reconsidered.

A check for outliers was done by first looking at the maximum and minimum values of the input data. These were concluded to be reasonable which suggests that the input data didn't have any outliers. A plot of the input data was also analyzed and no clear outliers were found. A test if there was any trend in the input modeling was done through the *testMean* function. This resulted in a one, which as mentioned previously, indicates that the null hypothesis should be discarded. With the same reasoning as before, it was determined that it may not be beneficial to incorporate the trend into the model so a model without this property would be created first, and if that was not satisfactory, the trend would be revisited.

2.2.3 Estimating a model for the input

The first step is to find the orders for the A2- and B-polynomials. To do this a model first needs to be estimated for the input, see Equation 5. When examining the ACF of the model data net radiation a clear season of about 24 could be viewed. This indicates that a differentiation may be beneficial in handling the data set. It was decided to differentiate the data and a differentiation with `[1 zeros(1,23) -1]` was done which created a new dataset, called w_t. Using the function *estimateARMA*, an ARMA model was estimated for the differentiated dataset. This function was used by starting off with as few parameters as possible and then adding one by one. The ACF and PACF plots were observed to decide which parameters to add.

$$A_3(z)x_t = C_3(z)w_t \qquad \text{Equation 5}$$

The estimated model had the A- and C-polynomials:

A(z) = 1 + 0.05284 (±0.03745) z^-1 - 0.1012 (±0.03679) z^-2 + 0.2068 (±0.03789) z^-3 + 0.5606 (±0.02286) z^-24

C(z) = 1 + 0.8191 (±0.04665) z^-1 + 0.5513 (±0.06221) z^-2 + 0.5943 (±0.06603) z^-3 + 0.4171 (±0.05822) z^-4 + 0.2914 (±0.0519) z^-5 + 0.2071 (±0.04621) z^-6 + 0.1534 (±0.04043) z^-7 + 0.1111 (±0.03673) z^-8 + 0.0381 (±0.02828) z^-9

After estimating a model that gave white noise for the differentiated data the differentiation was added to the model.

2.2.4 Prediction of input

When a model for the input had been estimated a prediction of the input based on this model was created. The k-step prediction is made using Equation 1, where G_k is calculated using the Diophantine equation, see Equation 2.

The prediction error which is the difference between the real modeling input data and the predicted data is white which means the model is working well. In the ACF and PACF plots, from the function *plotPlots*, there are still peaks at 24 and 48 which means the seasonality has not been removed 100% but the model is still sufficient to move along. This is something one can come back to if necessary.

2.2.5 Testing the input model against validation and test data

The predicted model for the input data was first tested on the validation data and compared to the real data and a naive prediction. The naive prediction used a seasonality of 24 if the lag was above one. The predictions were plotted together for one, three and eighteen lags, see Figure 23–25. By observing the plot it can be seen that the predicted model follows the real data very well. The variance of the model residual for the validation data and lag 1 was 3540.57 and the variance for the naive prediction residual was 4185.47. For lag 3 these were 5285.95 for the predicted model and 9648.97 for the naive prediction and for lag 18 these were 5452.08 for the predicted model and 9648.97 for the naive predictor. The normalized variances for the predicted model were lower than for the naive predictor for lag 1, 3, and 18 as well, see Table 2. This implies the predicted model performs better than the naive prediction. The ACFs and PACFs are shown in Figure 26–28.

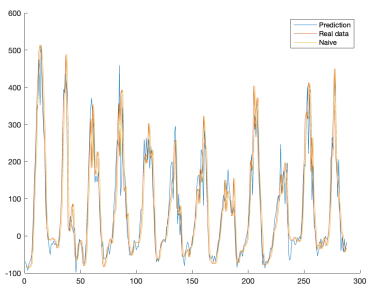

Figure 23: The model prediction, naive prediction, and true data for the one-step prediction of the validation data plotted together.

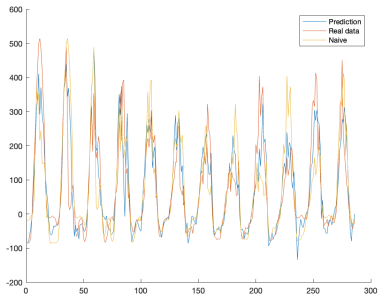

Figure 24: The model prediction, naive prediction, and true data for the three-step prediction of the validation data plotted together.

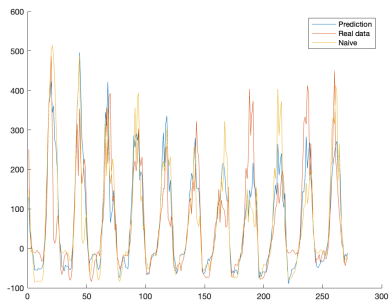

Figure 25: The model prediction, naive prediction, and true data for the eighteen-step prediction of the validation data plotted together.

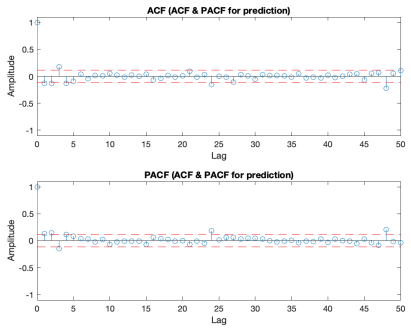

Figure 26: The autocorrelation function for the model one-step prediction residual for the validation data set

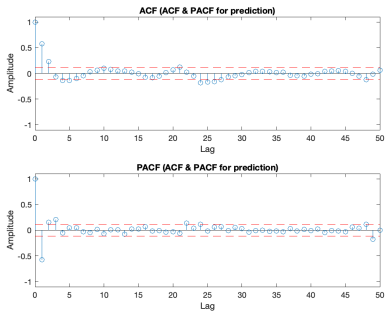

Figure 27: The autocorrelation function for the model three-step prediction residual for the validation data set

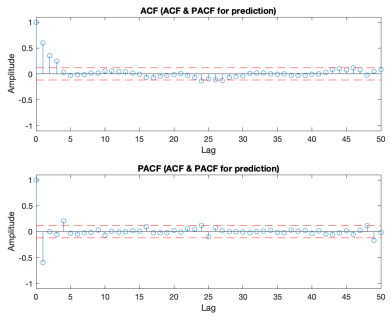

Figure 28: The autocorrelation function for the model eighteen-step prediction residual for the validation data set

The predicted model was also tested on the test data at time 4680 to time 4847. This prediction was compared against the real input data and the naive prediction by first plotting all of the data sets in the same graph, see Figure 29. The naive prediction used a seasonality of 24 for lags higher than 1. The predicted data follows the real data well. The variance of the model residual for the validation data for lag one was 1117.38 and the variance for the naive prediction residual was 3478.67. For lag three these were 2463.99 for the predicted model and 4514.01 for the naive prediction and for lag 18 these were 2334.22 for the predicted model and 4514.01 for the naive predictor. When using the test data the normalized variances were also lower for the predicted model than for the naive predictor for lag 1, 3, and 18, see Table 2. All these lead to the conclusion that the predicted model worked better than a naive prediction which is the aim.

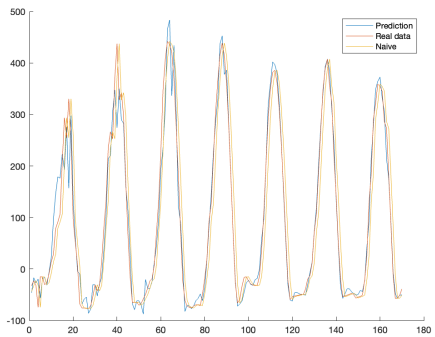

Figure 29:The real data, prediction and naive prediction of the input test data for lag 1

Table 2: The normalized variances of the predicted model and the naive predictor on the validation data and test data

Scenario:	*k=1 for the validation data*	*k=3 for the validation data*	*k=18 for the validation data*	*k=1 for the test data*	*k=3 for the test data*	*k=18 for the test data*
Normalized variance from the model:	0.19	0.28	0.34	0.04	0.09	0.08
Normalized variance from naive predictor:	0.22	0.51	0.61	0.12	0.16	0.16

2.2.6 Model order selection

To be able to find the orders of A2 and B the cross-correlation between the input and the output filtered through the model just found was observed. The input model was filtered through the model's A- and C-polynomials according to Equation 6 to get ω_t and the output model was filtered through the same as the input, see Equation 7, to get ε_t.

$$A_3(z)x_t = C_3(z)\omega_t \qquad \text{Equation 6}$$

$$\frac{A_3(z)}{C_3(z)} y_t = \epsilon_t \qquad \text{Equation 7}$$

By looking at this cross-correlation , see Figure 30, the orders (d, r, s) can be decided. The d is the delay from zero. The first peak that is above the confidence interval is at zero which means there is no delay between input and output. Therefore the d should be zero. If r is zero it means there are a finite number of peaks outside of the confidence interval which can be observed in the figure. The cross-correlation should go to zero, or inside the confidence interval, at lag d+s. In the figure, it can be observed the cross-correlation is zero first at four which indicates that s could be four. This led to the conclusion that the orders were (0,0,4).

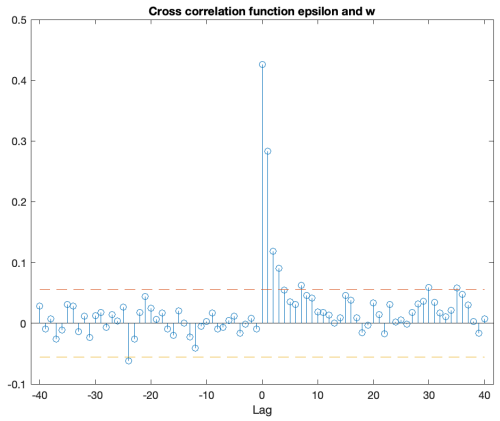

Figure 30: The cross-correlation between filtered input and filtered output

These orders were used to decide the A2 and B polynomials. From the decided orders the A2 polynomial was set as [1] and the B polynomial was set as [1 1 1 1]. Then y_t was modeled as a function of x_t, the input using the Matlab function *pem*. A check for pole-zero cancellation was done by plotting the poles and zeros for both this model and the input model and observing if the poles and zeros overlapped. No overlapping was observed and therefore there was no pole-zero cancellation.

2.2.7 ARMA process in Box Jenkins

The next step was to estimate the A1 and C polynomials in Equation 8.

$$\bar{e}_t = \frac{C_1(z)}{A_1(z)} e_t \qquad \text{Equation 8}$$

To do this the residual $\tilde{e}_t$ was estimated using the B and A_2 polynomial estimated earlier. This was done using Equation 9.

$$\bar{e}_t = y_t - \frac{\widehat{B}(z) z^{-\hat{d}}}{\widehat{A_2}(z)} x_t \qquad \text{Equation 9}$$

After the residual was estimated the A1 and C1 polynomials were estimated using the estimateARMA function. A and C parameters were added one by one and the ACF and PACF were observed between each addition to decide what to add next. The resulting model had the A and C polynomials:

A(z) = 1 - 0.5654 (±0.02188) z^-1 - 0.00126 (±0.01295) z^-7 - 0.4027 (±0.01968) z^-23
C(z) = 1 - 0.01889 (±0.02197) z^-1 + 0.7329 (±0.02863) z^-3 + 0.132 (±0.02006) z^-5 + 0.04535 (±0.02921) z^-6

2.2.8 Predicting the Box Jenkins model

A Box Jenkins model was estimated using the model orders for A1, A2, B, and C, see Table 3. This was done using the *pem* function.

Table 3: The orders for the different polynomials

Polynomials
A1 = [1 1 0 1 0 1 1]
A2 = [1]
B = [1 1 1 1]
C = [1 1 0 0 0 0 0 1 zeros(1,15) 1]

A check for pole-zero cancellation was done here as well on the estimated Box Jenkins model by plotting the poles and zeros. No pole-zero cancellation was found.

The prediction for the Box Jenkins model was built by using the estimated model's polynomials in Equation 4. In this equation the predicted input, real input and output is used to create the predicted output. The final model had the polynomials:

B(z) = 0.00625 (+/- 0.0002288) + 0.00420 (+/- 0.0002401) z^-1 + 0.00182 (+/- 0.0002399) z^-2 + 0.00105 (+/- 0.0002276) z^-3

C1(z) = 1 - 1.698 (+/- 0.1172) z^-1 + 1.346 (+/- 0.1655) z^-2 - 1.333 (+/- 0.1631) z^-3 + 1.178 (+/- 0.1348) z^-4 - 0.1203 (+/- 0.09701) z^-5 - 0.3175 (+/- 0.08824) z^-6 + 0.1102 (+/- 0.08502) z^-7 - 0.1688 (+/- 0.08511) z^-8 + 0.07483 (+/- 0.08558) z^-9 - 0.1028 (+ /- 0.08502) z^-10 + 0.1213 (+/- 0.08649) z^-11 - 0.1267 (+/- 0.08604) z^-12 + 0.1316 (+/- 0.08718) z^-13 - 0.1172 (+/- 0.08628) z^-14 + 0.02185 (+/- 0.08637) z^-15 + 0.0567 (+/- 0.08572) z^-16 - 0.05738 (+/- 0.08567) z^-17 + 0.09134 (+/- 0.08512) z^-18 - 0.09669 (+/- 0.08569) z^-19 + 0.04958 (+/- 0.08089) z^-20 - 0.003307 (+/- 0.07201) z^-21 + 0.06523 (+/- 0.06191) z^-22 - 0.03656 (+/- 0.03613) z^-23

A1(z) = 1 - 3.056 (+/- 0.1136) z^-1 + 4.086 (+/- 0.2981) z^-2 - 3.931 (+/- 0.3872) z^-3 + 3.692 (+/- 0.3576) z^-4 - 2.554 (+/- 0.2491) z^-5 + 0.7638 (+/- 0.09512) z^-6

A2(z) = 1

2.2.9 Evaluating the prediction

Lastly, the prediction of the output was evaluated. First a prediction on the validation data was done and evaluated. The predicted output with lag one, three and eighteen was plotted together with a naive prediction and the real data, see Figures 31–33. The naive prediction used a season of 24 if the lag was higher than one. The variance for the model residual with lag one was 0.51 and the variance of the naive prediction residual was 0.73. With lag three the variance of the model residual was 2.00 and

the variance of the naive prediction residual was 5.30. For lag 18 the variance of the model residual was 5.61 and for the naive prediction, it was 5.31. The normalized variance for the model prediction was 0.07, 0.,26 and 0.72 for lag 1, 3 and 18 respectively and the normalized variance for the naive prediction was 0.1, 0.69 and 0.68. The ACF and PACF for each lag were also observed, see Figures 34–36 and they were all similar to a MA(k-1) process which signifies the model works relatively well.

Both the normalized variance and the unnormalized variance for the predicted model at lag one and three were lower than the ones for the naive predictor. This implies the predicted output model performs better than the naive predictor at lower lags. At lag 18 the naive predictor performs better than the predicted model, however only slightly. Since the predicted model performed well at low lags it was decided it was a sufficient model and no changes were made.

The normality of the ACF and the PACF for each of these cases was also checked and they weren't normally distributed in any of the cases.

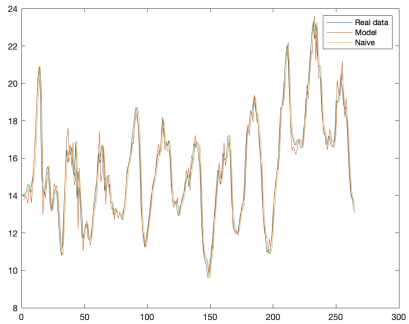

Figure 31: The model prediction, naive prediction, and true data for the one-step prediction of the validation data plotted together.

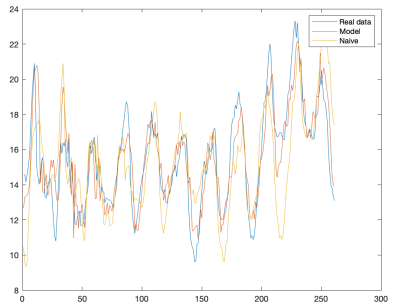

Figure 32: The model prediction, naive prediction, and true data for the three-step prediction of the validation data plotted together.

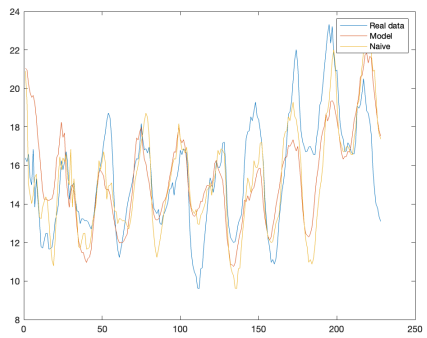

Figure 33: The model prediction, naive prediction, and true data for the eighteen-step prediction of the validation data plotted together.

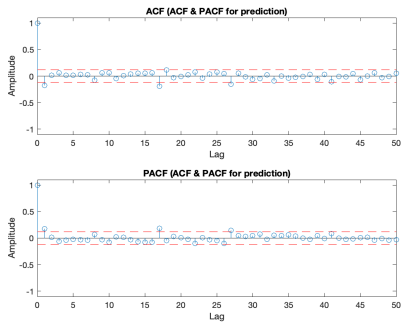

Figure 34: The autocorrelation function for the model one-step prediction residual for the validation data set.

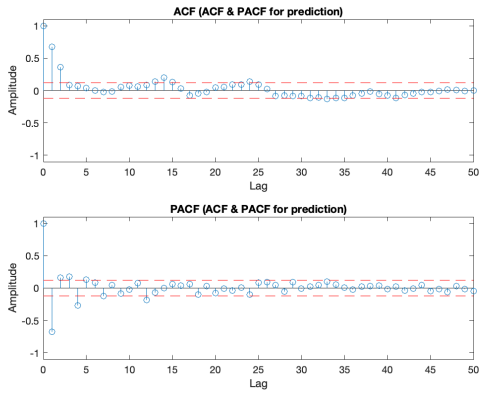

Figure 35: The autocorrelation function for the model three-step prediction residual for the validation data set.

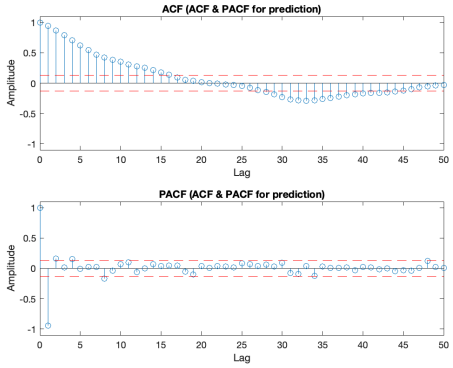

Figure 36: The autocorrelation function for the model eighteen-step prediction residual for the validation data set.

To further validate the model, it was tested on two different test data sets. The data sets used were the same as for the input. The prediction for these test sets were plotted against the real data and a naive prediction for lag 1, 3 and 18, see Figures 37–42. The naive prediction had a season of 24 for lags higher than 1. The ACF and PACF for both test sets at lag 1, 3 and 18 were also plotted, see Figures 43–48. By observing the ACF and PACF it was concluded the prediction followed a MA(k-1) process well for testdata 2. For testdata 1 the model didn't perform as well and the residual isn't white for lag 1. The normalized variance and variance were also calculated for the two test sets as well as for the naive prediction, see table 4. It can be noted that the predicted model performs worse than the naive prediction in some cases of higher lags but for lower lags the predicted model outperforms the naive prediction.

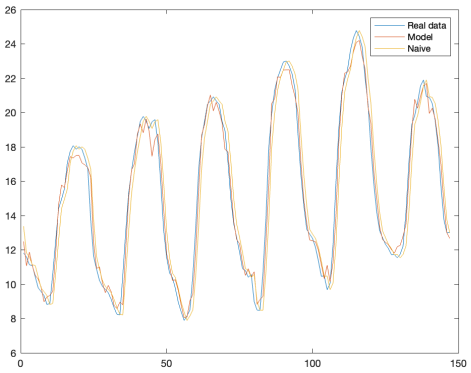

Figure 37: The model one-step prediction compared to the test data and the naive predictor for the first test data set.

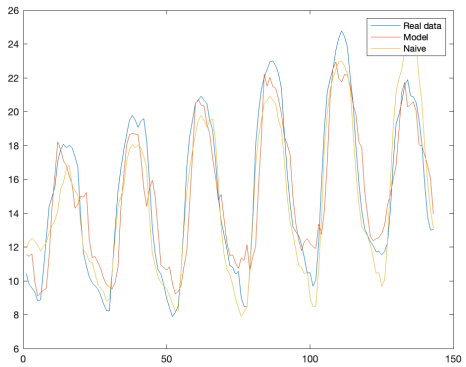

Figure 38: The model three-step prediction compared to the test data and the naive predictor for the first test data set.

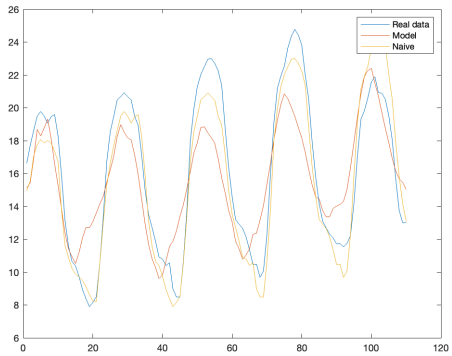

Figure 39: The model eighteen-step prediction compared to the test data and the naive predictor for the first test data set.

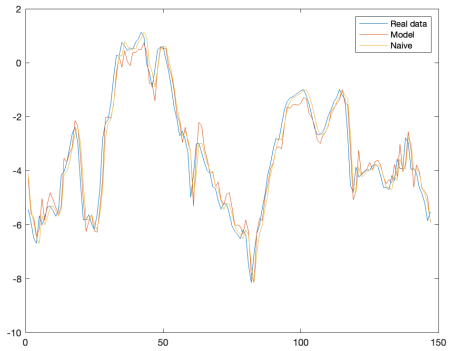

Figure 40: The model one-step prediction compared to the test data and the naive predictor for the second test data set.

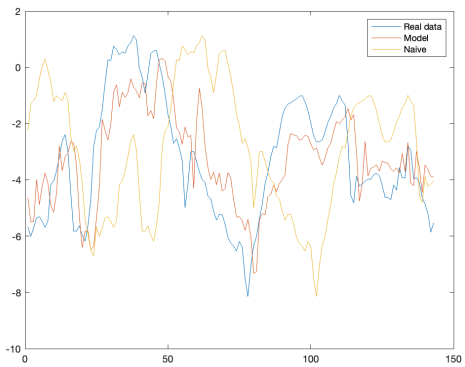

Figure 41: The model three-step prediction compared to the test data and the naive predictor for the second test data set.

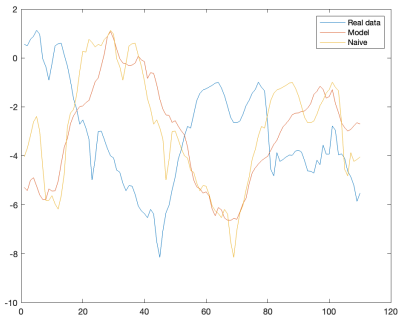

Figure 42: The model eighteen-step prediction compared to the test data and the naive predictor for the second test data set.

The variance of the model residual and the naive prediction residual was calculated for lag 1, 3, and 18, together with the normalized variance for each of these, see Table 4.

Table 4: The variances and normalized variances of the predicted model and the naive predictor on the test data sets for the Kalman filter.

Scenario:	*k=1 for the first test data*	*k=3 for the first test data*	*k=18 for the first test data*	*k=1 for the second test data*	*k=3 for the second test data*	*k=18 for the second test data*
Variance from the model:	0.54	4.08	7.36	0.32	2.00	14.92
Variance from naive predictor:	1.42	6.46	6.46	0.32	16.41	16.41
Normalized variance from the model:	0.02	0.18	0.32	0.07	0.43	3.37
Normalized variance	0.06	0.29	0.28	0.07	3.55	3.70

from naive predictor:						

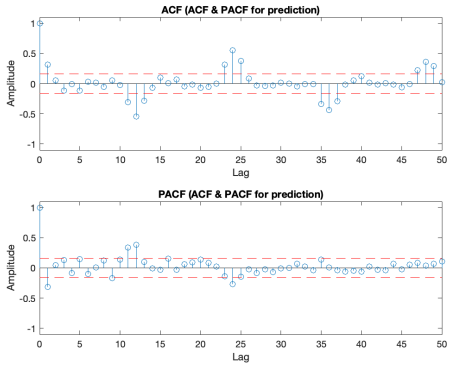

Figure 43: The autocorrelation function for the model one-step prediction for the first test data set.

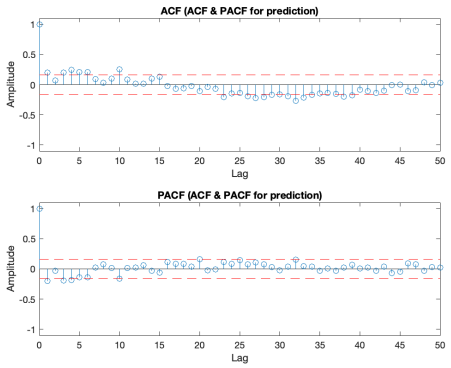

Figure 44: The autocorrelation function for the model one-step prediction for the second test data.

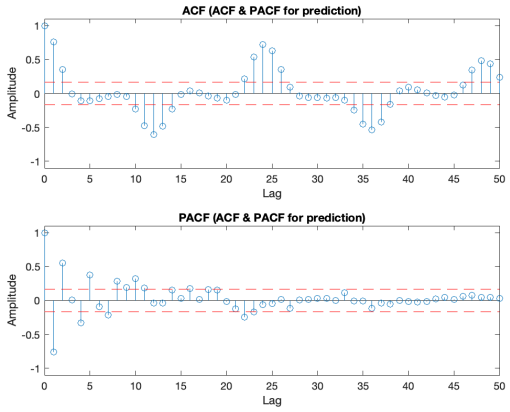

Figure 45: The autocorrelation function for the model three-step prediction for the first test data set.

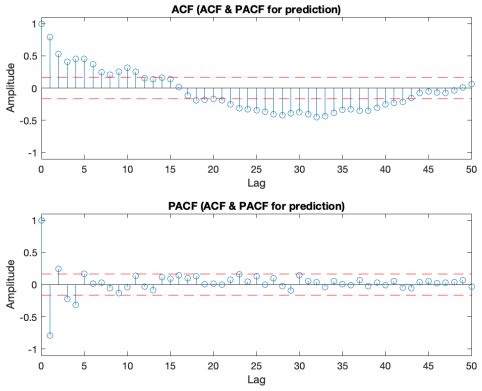

Figure 46: The autocorrelation function for the model three-step prediction for the second test data set.

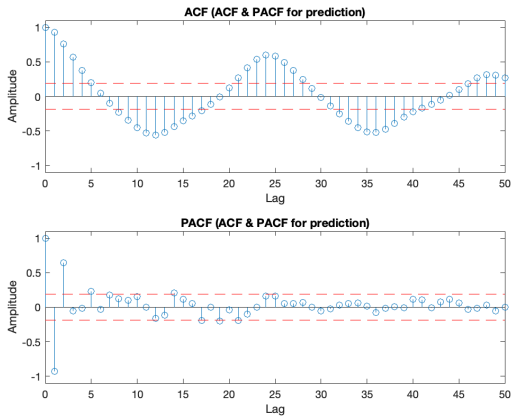

Figure 47: The autocorrelation function for the model eighteen-step prediction for the first test data set.

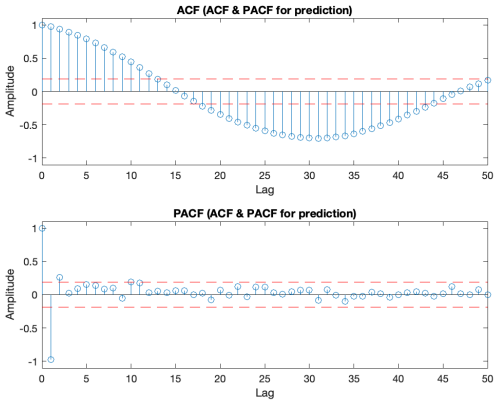

Figure 48: The autocorrelation function for the model eighteen-step prediction for the second test data set.

2.2.10 Stand-alone Predictor

A stand-alone predictor was created for part B. The predictor is found in the file *predCodeB_grp008* and includes a prediction of input as well as output using the parameters estimated. It returns the predicted values for the output using the model. The stand-alone predictor was tested with the *examCode* file and called upon in the code.

2.2.11 The model

For most of the data sets tested the predicted model for the output performed effectively. The model performed worse for higher lags which is expected since it is more difficult to predict further into the future. To create a prediction for output using input can be difficult since the input prediction needs to work well for the whole model to predict well. In this case the input model worked relatively well which should improve the predicted model performance. Since the output model performed better than the naive for lower lags it was deemed unnecessary to change the input model even though it had some small peaks at 24.

2.3 Part C

In part C of this project, a recursive predictor of the model in 2.2 was constructed. This predictor was initiated using the modeling data but was allowed to update its parameters using the data made available as time passes by.

2.3.1 Overview of the Data

There was no additional data for this task and thus, there was no need for a rigorous control of the data properties as in 2.1 and 2.2. The data was instead set up directly based on the selected timelines and traits from previous parts of the project.

2.3.2 Kalman Filter of Input Data

Although it might be considered cheating to predict the whole input using Kalman before predicting the output, dividing the Kalman filtering up into two parts, one for the input and one for the output, was seen as a practical way to simplify the code and recommended by the teaching staff. Therefore, this was done. However, it was noted that this would not be possible in reality since the whole input data wouldn't be available before predicting the output. The first Kalman filter was set up, following Equations 10 and 11, with the net radiation data as the output to be predicted. The model from part B was used to build the filter. The orders of the ARMA model from the input in part B were used to form the matrices in the filter. The coefficients of the model were used to create the initial estimates of the parameter values.

$$\hat{x}_{t+1|t} = A\hat{x}_{t|t} + Bu_t \qquad \text{Equation 10}$$

$$\hat{y}_{t+1|t} = C\hat{x}_{t+1|t} \qquad \text{Equation 11}$$

After the model had been set up with initial values, the filter was ran through. The first values were dismissed so that the index of a matrix would not be below one, but the loop ran as for the rest of the input data set except the last few values that would create similar difficulties. After this, the values were updated according to the theory for Kalman filters. In the Kalman filter the prediction equations are rewritten to allow for quicker recalculations as new data is made available. The parameters are then updated according to this new information based on the Kalman gain, K_t. For a lag of one, the

update was given in the loop, but for larger lags a second loop is present to calculate the predictions. These values are then stored in other variables.

The prediction error of the radiation data was then calculated and a Monti test conducted and *plotPlots* plotted. The whiteness test was positive and the ACF and PACF looked reasonably white. The ACF and PACF can be viewed in Figure 49. However, it is worth mentioning that they are not normally distributed which affects with what certainty the formulas can be applied. The parameter estimations were plotted and compared to the parameters for part B, which can be observed in Figure 50. These values differed quite a bit as can be seen in the figure.

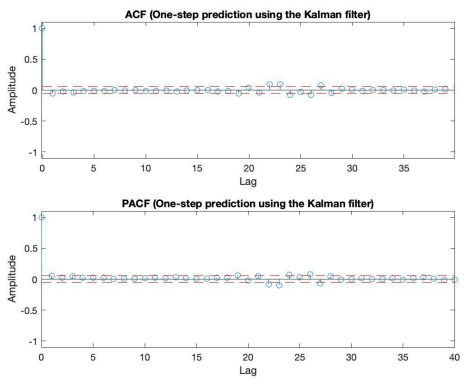

Figure 49: The ACF and PACF of the input modeling data residual.

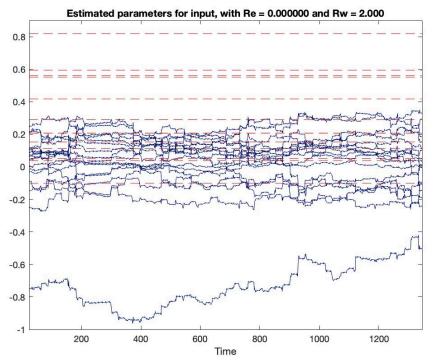

Figure 50: The parameter estimations with confidence intervals according to the Kalman filter and according to the values obtained in part B

2.3.3 Evaluating the Kalman Filter of the Input Data

As the Kalman filter worked on the modeling data, it was time to try it on the validation data. The A- and C-polynomials in the end of the validation data were:

A(z) = 1 - 0.511(±0.00268) z^-1 - 0.154 (±0.00313) z^-2 + 0.287(±0.00206) z^-3 - 0.146 (±0.00150) z^-24 - 0.0968(±0.00175) z^-25 + 0.0291 (±0.00175) z^-26 - 0.0253(±0.00158) z^-27 - 0.293 (±0.00148) z^-48

C(z) = 1 - 0.183(±0.00263) z^-1 - 0.0159(±0.00232) z^-2 + 0.329 (±0.00191) z^-3 + 0.119(±0.00179) z^-4 + 0.0524(±0.00172) z^-5 + 0.0334 (±0.00167) z^-6 + 0.0419(±0.00163) z^-7 + 0.0480(±0.00161) z^-8 + 0.00817 (±0.00160) z^-9

The evaluation was done by plotting the true data with the model prediction and the naive prediction. The result can be seen in Figure 51. The ACF and PACF were examined as well, which can be seen in Figure 52. The ACF was not normally distributed here either. This residual did not pass the whiteness test. The variance of the prediction residual was 7 827.76 and the normalized variance was 0.32. This model was not as promising as the models had been in the past. Because of this, the covariance matrices for the state noise and for the white noise w were changed to see if any improvement could be seen. The covariance matrix for the initial values were also changed but no improvement by any of these changes were observed. However, since the first model looked reasonable in the plot, it was decided to proceed and check the test data and output model before making further adjustments to the input model.

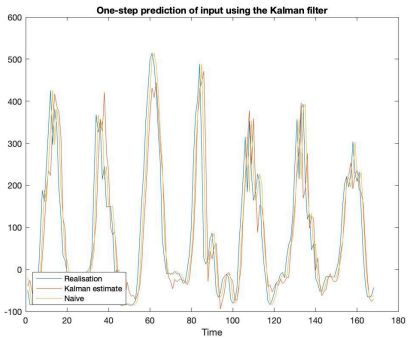

Figure 51: The naive prediction, model prediction and true data for the one-step prediction of the validation data.

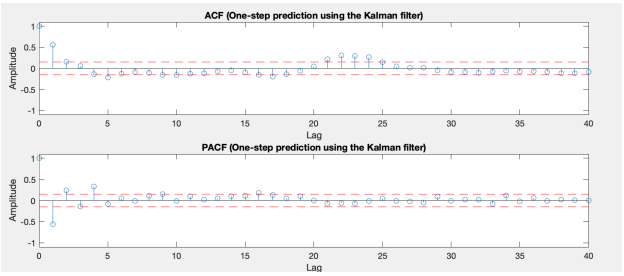

Figure 52: The autocorrelation function and autocorrelation function for the one-step prediction of the validation data.

The test data was also checked, and for the first test data the residual was deemed white noise. The ACF and PACF are found in Figure 53. The prediction along with the naive prediction and the true values is found in Figure 54. The residual variance was 2630.49 in this case and the normalized residual variance was 0.18.

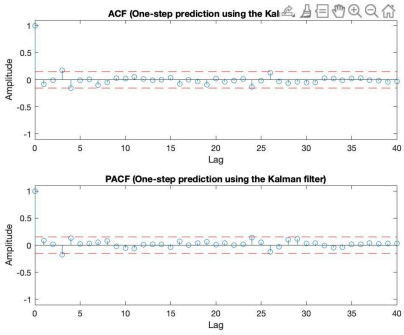

Figure 53: The autocorrelation function and autocorrelation function for the one-step prediction of the first test data.

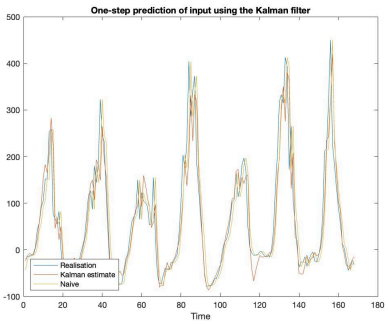

Figure 54: The naive prediction, model prediction and true data for the one-step prediction of the first test data.

For the second test data, the residual was also deemed white noise. The ACF and PACF are found in Figure 55. The prediction along with the naive prediction and the true values is found in Figure 56. The residual variance was 161.49 and the normalized variance was 0.27.

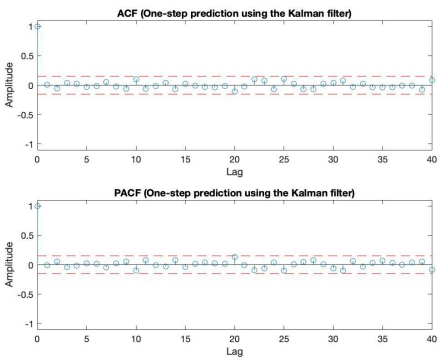

Figure 55: The autocorrelation function and autocorrelation function for the one-step prediction of the second test data.

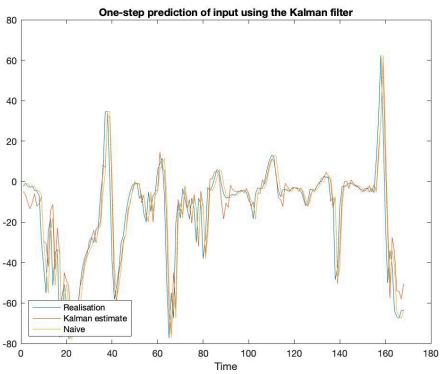

Figure 56: The naive prediction, model prediction and true data for the one-step prediction of the second test data.

Although better results for the validation data would be better, since the plot still looked reasonable and the results were very promising for the test data, it was decided to continue using the model for the input. If difficulties would arise later it would be appropriate to revisit this model.

2.3.4 Setting up the Output Data Model

As the input data had been dealt with, it was time to set up the model to estimate the temperature. This model was set up according to the ARMAX model that was obtained in 2.2. These values of the parameters from the ARMAX model were used as the initial parameters for the Kalman filter, in the same way as 2.3.2. This Kalman filter was set up so that x was the radiation data and y was the temperature data. Here the polynomials K_A, K_B and K_C defined according to Equations 12–14 were formed.

$$K_A = A_1(z) * A_2(z) \qquad \text{Equation 12}$$

$$K_B = A_1(z) * B(z) * z^{-d} \qquad \text{Equation 13}$$

$$K_C = A_2(z) * C_1(z) \qquad \text{Equation 14}$$

2.3.5 Kalman Filter of Output Data

This Kalman filter was constructed in a similar way to the previous one. The difference was that this filter had an input. The primary impact the input has is that the C-matrix changes drastically. Otherwise, the structure is very similar with initializing the variables and creating a loop with the appropriate time period that starts with the one-step prediction. Inside the loop, there is another loop if the time lag is larger than one.

To check the functionality of this Kalman filter, the prediction was plotted against the true values and the naive prediction. This can be seen in Figure 57. The residual was also calculated and its whiteness and other traits were evaluated. This all looked promising. The estimated parameters can be seen in Figure 58.

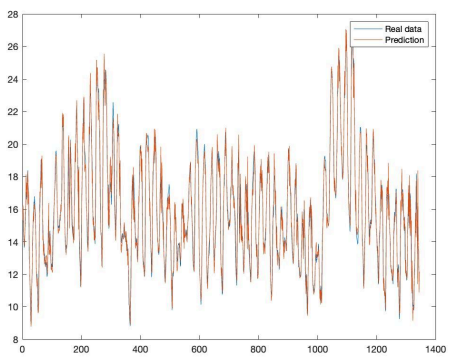

Figure 57: The model prediction, naive prediction, and true data for the one-step prediction of the model data plotted together.

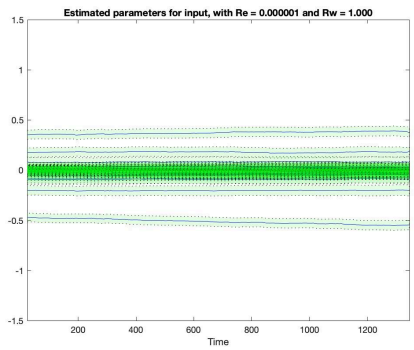

Figure 58: The estimated model parameters for the output Kalman filter.

2.3.6 Evaluating the Kalman Filter of the Output Data

The next step was to evaluate the Kalman filter by seeing how well it works for the validation data. The polynomials at the end of the validation can be seen in Table 5. The model done by the Kalman

filter was evaluated by creating the naive prediction and plotting the naive prediction, the real data, and the model prediction together for this data set. These plots can be seen in Figures 59 through 61. The model residual was also determined and a Monti whiteness test was conducted on it that deemed it to be white noise. The residual variance for the one-step prediction of the validation data was 0.82 and for the naive predictor, it was 1.96. The normalized variances were 0.16 and 0.39, respectively. For the three-step prediction the residual variance was 5.04, compared to the naive prediction at 24.61. The normalized variances were 0.99 and 4.86, respectively. For the eighteen step prediction, the prediction variance was 4.23 compared to the naive prediction's 24.61. The normalized variances were 0.84 and 4.86. The ACF and PACF were examined and can be viewed in Figures 62 through 64.

Table 5: The parameter's final values in the validation data

KA	Standard deviation	KC	Standard deviation
-0.555	0.048975576	0.359	0.048132207
-0.203	0.054790345	0.19	0.049971643
-0.0703	0.055387668	0.0939	0.050060511
-0.0355	0.055337728	0.00926	0.049923537
-0.0427	0.054375958	0.0227	0.049452706
-0.0956	0.0481184	-0.0283	0.045092672
KB	Standard deviation	-0.0387	0.044558745
0.00564	0.00459535	-0.0346	0.044487049
-0.000774	0.004919337	-0.0475	0.044169757
-0.00238	0.004857751	-0.0528	0.043979689
-0.00016	0.004837319	-0.0248	0.043912494
-0.00146	0.004834403	0.00916	0.043970366
0.0012	0.004721707	0.0145	0.043841418
-0.000694	0.004524996	0.0273	0.043749526
-0.00173	0.00430518	0.0453	0.043847917
-0.00347	0.004091355	0.0367	0.044092907
-0.000333	0.003344636	0.0112	0.044303284
		0.0202	0.044474738
		0.00239	0.04452241
		-0.0215	0.044597051
		-0.0516	0.044571393
		-0.043	0.044862491
		-0.036	0.044614098

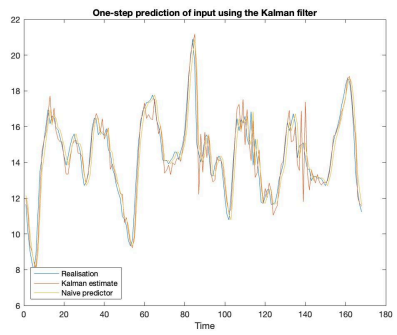

Figure 59: The model prediction, naive prediction, and true data for the one-step prediction of the validation data plotted together.

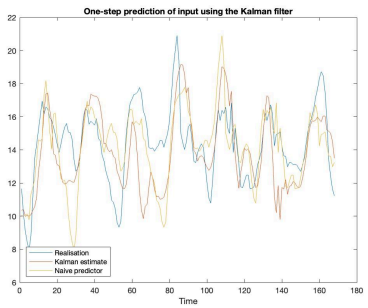

Figure 60: The model prediction, naive prediction, and true data for the three-step prediction of the validation data plotted together.

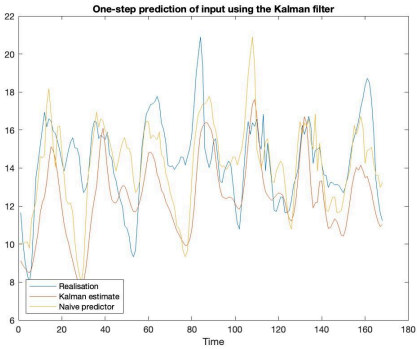

Figure 61: The model prediction, naive prediction, and true data for the eighteen-step prediction of the validation data plotted together.

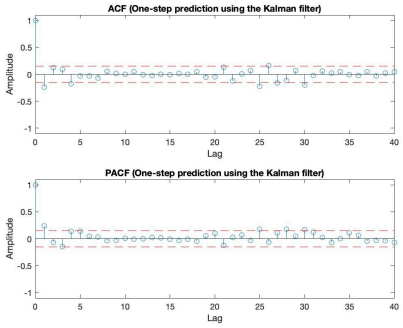

Figure 62: The autocorrelation function and partial autocorrelation function for the one-step prediction of the validation data.

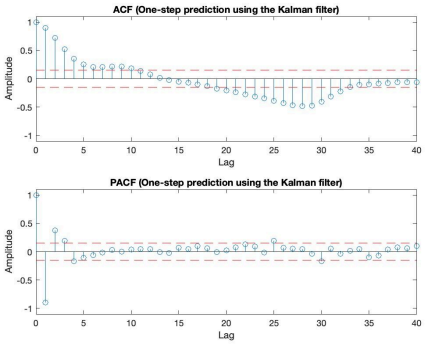

Figure 63: The autocorrelation function and partial autocorrelation function for the three-step prediction of the validation data.

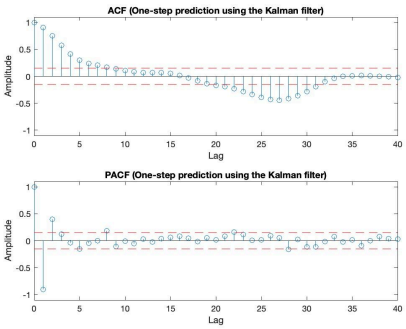

Figure 64: The autocorrelation function and partial autocorrelation function for the eighteen-step prediction of the validation data.

These results show that the Kalman filter gives a well-fitting prediction of the data and that it is superior to that of the naive predictor. The variance of the validation data set is 5.06, which is substantially larger than that of the prediction residuals.

When it worked well with the validation data it was tested with the test data instead. For the first set of test data, the week after the validation data, the residual was deemed white noise in the Monti test. However, the ACF was not normally distributed. The plots of the naive prediction, model prediction, and true data can be observed in Figures 65 through 67. The variance for the one-step model prediction residual for the first test data set was 0.40. The same value for the naive predictor was 1.22. For the three-step prediction, the residual was 2.96, compared to the naive prediction at 23.03. For the eighteen-step prediction the variance was 5.26 and for the naive prediction it was 23.03. The normalized variances are found in Table 6. The ACFs and PACFs are plotted in Figures 68 through 70. The variance of the data set was 8.98.

Table 6: The normalized variances of the predicted model and the naive predictor on the test data sets for the Kalman filter.

Scenario:	*k=1 for the first test data*	*k=3 for the first test data*	*k=18 for the first test data*	*k=1 for the second test data*	*k=3 for the second test data*	*k=18 for the second test data*

Normalized variance from the model:	0.04	0.33	0.59	0.05	1.22	1.93
Normalized variance from naive predictor:	0.14	2.57	2.57	0.08	3.85	3.85

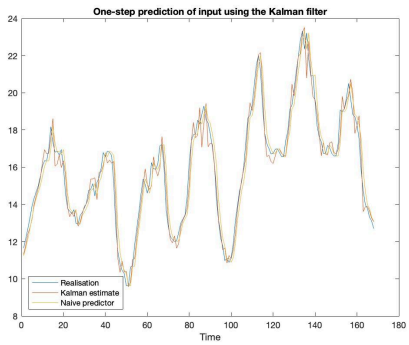

Figure 65: The model prediction, naive prediction, and true data for the one-step prediction of the first test data plotted together.

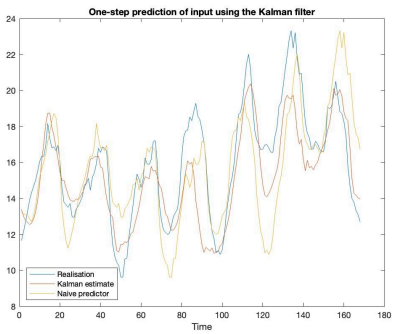

Figure 66: The model prediction, naive prediction, and true data for the three-step prediction of the first test data plotted together.

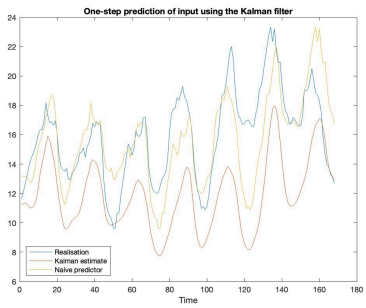

Figure 67: The model prediction, naive prediction, and true data for the eighteen-step prediction of the first test data plotted together.

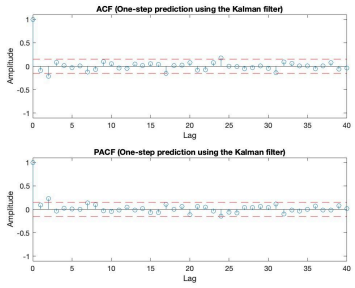

Figure 68: The autocorrelation function and partial autocorrelation function for the one-step prediction of the first test data.

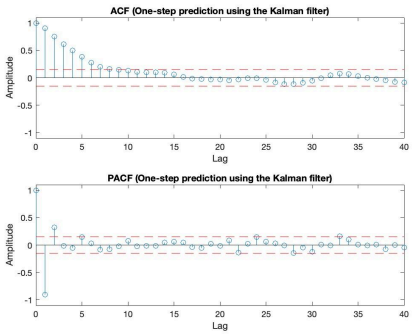

Figure 69: The autocorrelation function and partial autocorrelation function for the three-step prediction of the first test data.

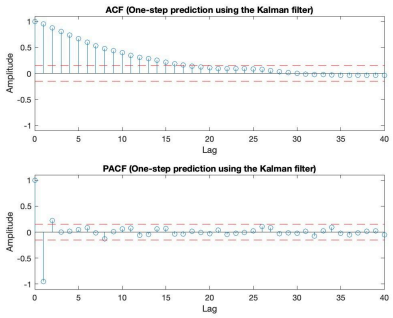

Figure 70: The autocorrelation function and partial autocorrelation function for the eighteen-step prediction of the first test data.

Next, the second test data set was tested. The residual was deemed white noise in the Monti test. However, the ACF was not normally distributed. The plots of the naive prediction, model prediction, and true data can be observed in Figures 71 through 73. The variance for the one-step model prediction residual for the first test data set was 0.25. The same value for the naive predictor was 0.38. For the three-step prediction, the residual was 5.72, compared to the naive prediction at 17.98. The eighteen-step prediction residual had a variance of 9.03 compared to the naive predictor that had a variance of 17.98. The ACFs and PACFs are plotted in Figures 74 through 76. The variance of the data set was 4.68.

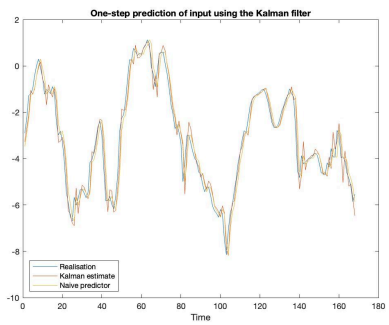

Figure 71: The model prediction, naive prediction, and true data for the one-step prediction of the second test data plotted together.

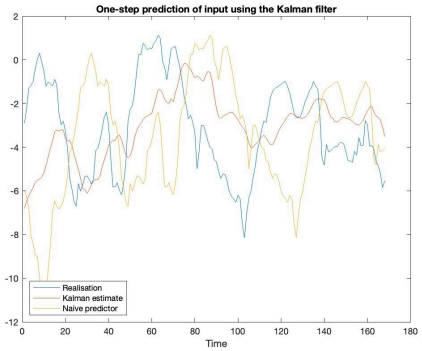

Figure 72: The model prediction, naive prediction, and true data for the three-step prediction of the second test data plotted together.

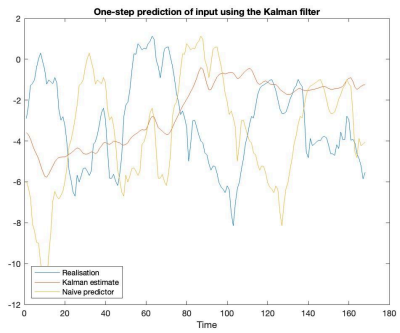

Figure 73: The model prediction, naive prediction, and true data for the eighteen-step prediction of the second test data plotted together.

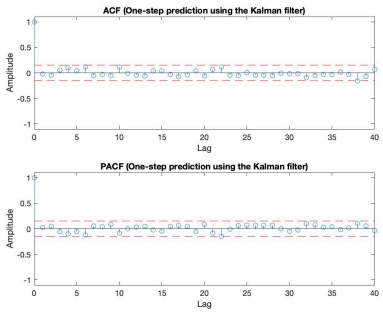

Figure 74: The autocorrelation function and partial autocorrelation function for the one-step prediction of the second test data.

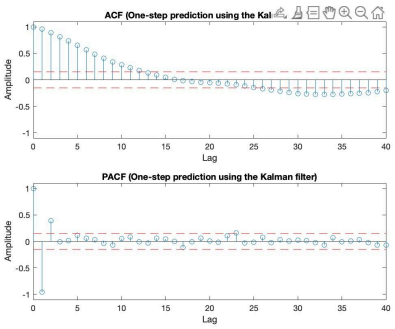

Figure 75: The autocorrelation function and partial autocorrelation function for the three-step prediction of the second test data.

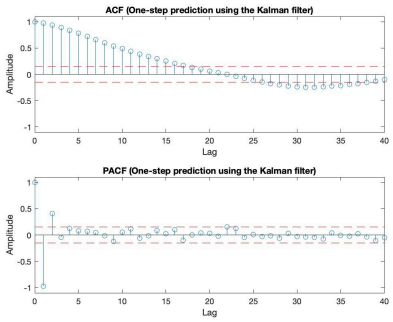

Figure 76: The autocorrelation function and partial autocorrelation function for the eighteen-step prediction of the second test data.

The model performs very well in both of these tests, which is a good sign that the model works well and can predict the temperature for other times reasonably well.

2.3.7 Is it Possible to Remove Parameters from the Model?

To get an understanding of whether it is possible to remove parameters from the model or not, it was first determined which parameter was the most likely to be able to be removed. The coefficient that was the closest to zero after the estimation through the Kalman filter was the tenth element which has a value of -0.0050. This was chosen as the parameter to try to remove. The code for part C was copied into the file *projectCparameters*. There the tenth element was removed to see what the results would be. This led to a considerable worsening of the input validation data, but when keeping the output model constant the Monti tests for all of the data were still white. This means that it is possible to remove the tenth parameter and still have a robust model.

After examining the input model, it was deemed interesting to see if it would be possible to remove parameters from the output model as well. In part B, the parameter with the coefficient closest to zero is the fourth B-coefficient which is 0.0010. After removing this coefficient, the residuals still pass the whiteness test and the model still looks reasonable. This can be seen in Figure 77 which shows the one-step prediction without the two parameters. Similar checks were done on other lags as well. This indicates that it is possible to remove some of the parameters without significant loss of performance even though they were significant in part B.

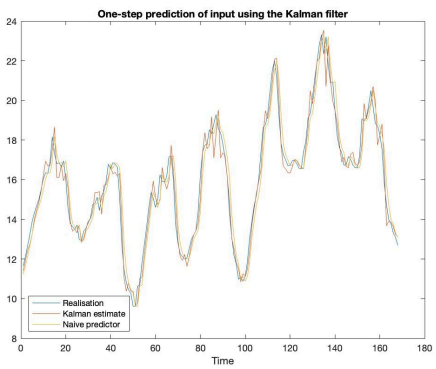

Figure 77: The plot of the one-step prediction using the Kalman filter with the model prediction after removing parameters, the naive prediction and the true values.

2.3.8 Stand-alone Predictor for Part C

When a solid model had been built, the stand-alone predictor was created. It is found in the file *predCodeC_grp008*. The stand-alone predictor is very similar to the process previously described. It was built so that it first ensures that the data inputs are correct, and otherwise either returns an error message or amends the data. After this, the Kalman filter uses all of the modeling data received, which is all data up to the one that is tested in the predictor, to estimate the parameters. The prediction of the radiation was then used as input into the second Kalman filter, in the same way as previously. After this was done the predicted temperatures for the right time period were returned. The stand-alone predictor was tested with the *examCode* file and called upon in the code for part C.

2.3.9 The model

This model has a vastly different structure than the two previous ones but uses the same data. This creates a good basis for understanding how the models work. The Kalman filter model leads to good results and all of the output residuals tested were deemed white noise. This means that the model manages to explain all movements in the data except what comes from pure noise. However, it is important to remember that this must be treated with caution since neither the data set nor the autocorrelation functions follow a normal distribution, which is assumed in the theory.

3. Analysis

Three different models for the same process have been produced, each with a unique set of tools. This section of the report explores how they compare, complement, and contrast with each other. The authors believe that the best way to set the stage is through a poem about the modeling procedure, and thus, here goes:

Beneath the shifting skies, we stand,
With data flowing, vast, unplanned.
A dance of numbers, seasons, light,
To glimpse tomorrow's hidden sight.

The clouds, they gather, winds will veer,
But models help us see, and hear:
A storm, a sunbeam, frost's embrace —
The rhythms nature leaves in place.

From moving averages to lagged insight,
Together, getting tomorrow right.
Autoregression whispers clear,
While trends and cycles reappear.

Through tests of fit and residual trace,
Your project ventures time's vast space.
From seasonality to error spread,
You map what lies ahead, unsaid.

~The authors with a little help from ChatGPT [OpenAI 2024]

With this reflection as the foundation, it is time to proceed with the analysis. It begins with a comparison of the models that showcases their respective strengths and limitations. Then, possible improvements are discussed.

3.1 Comparison

By looking at the normalized variances for the models in each part, the models can be compared. To do a fair comparison the normalised variance for the model residual on the two test sets were compared between each part. The normalized variances in part B were all larger or equal to the normalized variances for the same data and lag in part A which means the prediction in A performed better than the one in B. This contradicts the theory, saying the prediction in part B should perform better if the input has any correlation with the output. If the input doesn't explain anything in the output it would make the model worse. How well the output model performs depends on how good the input prediction is and a bad input prediction can worsen the model.

When comparing part B and C the normalized variances all increase for test data 1. On testdata 2 the normalized variances for lag 1 and 18 decrease but the normalized variance for lag 3 increase. This signifies that the model in part C performs better in some cases but worse in others. In theory the prediction in part C should perform better than the one in part B but this depends on how good the prediction for the inputs are.

The prediction in part C performs worse than the one in part A for most of the cases as well. The only case where the model in part C performs better is lag 18 on the second test set. This is reasonable since the model in part C should perform better at higher lags than the model in part A according to theory.

3.3 Possible Improvements

To create a good prediction model is difficult, especially for higher lags. One can sit hour after hour to try to improve a model, which was done in this project. The predicted models could of course be improved but many different model orders were tested in the different parts which were all worse then the final ones. Whenever issues were met in the project data was simulated to check that the code was working properly. This was to conclude if the issue was code related or model related.

The different test data were also working differently well with the different models. Some of the models predicted performed better on the first test data while some performed better on the second. This highlights the importance of using multiple different test sets to test the model. More test sets than the ones written in this report were tested on in the code to understand how well the predicted model worked.

4. Conclusions

In this project, three different models were constructed to predict the temperature in Tåstrup. One was only based on past measurements of temperature using an ARMA model, one on past measurements of temperature and radiation using an ARMAX model and the Box-Jenkins methodology, and the last implementing a Kalman filter to dynamically predict the temperature based on past measurements of temperature and radiation. They all did a reasonably well job at this, despite their different tool boxes.

It is however not possible to say that one model performed better or worse than the other models, since they predicted differently well between the predictions. This underscores that there are many different ways to predict data and they work varyingly. After all, if there was one best way, there would only be that way. However, the first model sticks out a bit as it has low normalized variance, but it also has its problems. Above all, the conclusion of this project is that predicting the future is difficult and that there are no promises.

5. *Sources*

Jakobsson, A. (2024). *Time Series Analysis Overview* [PowerPoint-slide].
OpenAI. (2024). *ChatGPT*. https://chat.openai.com/chat (Accessed 2024-12-14).
Poetry Foundation. (n.d). *Fire and Ice*. https://www.poetryfoundation.org/poems/44263/fire-and-ice [Accessed 2024-12-12]

6. *Appendix*

This report is a part of a zip-file which also contains the following documentation:

- *ProjectA*
- *ProjectB*
- *ProjectC*
- *predCodeA_grp008*
- *predCodeB_grp008*
- *predCodeC_grp008*
- *ProjectCparameters*
- *plotPlotsWillisLagerstedt*