VTT INFORMATION TECHNOLOGY

RESEARCH REPORT TTE1-2001-16

MODUS-project
WasteWater Case Study

Abnormality detection using SOM modeling

Version 1.1-1

18.12.2001

Mikko Hiirsalmi

Version history

Version	Date	Author(s)	Reviewer	Description
1.0-1	25.5.2001	Mikko Hiirsalmi		First version
1.1-1	8.6.2001	Mikko Hiirsalmi		Delivered to steering group 12.6.2001

Contact information

Mikko Hiirsalmi
VTT Information Technology
P.O. Box 1200, FIN-02044 VTT, Finland
Street Address: Tekniikantie 4 B, Espoo
Tel. +358 9 4561, fax +358 9 456 6027
Email: Mikko.Hiirsalmi@vtt.fi
Web: http://www.vtt.fi/tte

Last modified on 18 December, 2001
H:\MSJV3_2001_06073.doc

Abstract

We describe work aimed at applying neural network methods to detect abnormal conditions in quality measurements of an industrial chemical process. The methods predict the future values of the target variables based on multivariable histories of past sensor readings. A waste water cleaning plant using an activated sludge cleaning method served as our test environment for the methods.

Our aim has been to develop methods that may be implemented as online analysis tools at the process control environments and are capable of being interfaced with existing data sources. Previously we have tested Multi-Layered Perceptron networks on this problem and have described a demonstration system indicating how the measurements can be stored in a RapidBase main memory active database and how the neural network models may be used online for creating predictions for future values [Hii00a]. In this report we continue the previously reported analysis work with extended measurement data series and investigate especially the suitability of the Self-Organizing Map (SOM) architecture.

We have used a time-lagged delay line for embedding the temporal histories of the measurement time series and correlation analysis and statistical independency testing for detecting the most useful input variables for the analysis. Correlation analysis suggests that typically only the most recent measurements correlate with the future values.

In order to detect abnormal behaviour we have tought a SOM with data describing normally operating process. The learned map has been used to detect abnormalities in new data by monitoring the shortest distance of these new samples from the learned SOM codebook. Calibration data is used to select an optimal classification threshold. Experimental tests suggest that typically only short term predictions are possible in the case of the studied process. The best classification accuracies reach 80-90% levels.

Additionally we have used SOM to predict the future values by teaching it with a random sample of all the measurements and then creating an index from the SOM neurons to the list of teaching samples best matching that neuron. This index is used for nearest neighbour prediction while monitoring new data samples. In experimental tests the predictions were found to be promising for short intervals but lost accuracy when the prediction interval was made larger. The classification accuracies achieved were a little better than with the previously mentioned QE-monitoring results but the false positive rates were condiderably more severe.

Tiivistelmä

Käsittelemme laatukriteerien poikkeamien ennustamiseen tarkoitettujen hermoverkkomallien kehittämistyötä. Käytetyt menetelmät ennustavat tavoitemuuttujien tulevat arvot vanhojen mittausten monimuuttujaisiin aikasarjoihin pohjautuen. Menetelmien testiympäristönä on ollut teollisuuden biologista aktiivilieteprosessia käyttävä jätevesien puhdistuslaitos.

Tavoitteenamme on ollut kehittää menetelmiä, jotka voidaan toteuttaa tosiaikaisina analyysityökaluina prosessin valvontaympäristössä ja jotka voidaan liittää olemassa oleviin tietovarastoihin. Aiemmin olemme kokeilleet monikerros Perceptron hermoverkkoarkkitehtuuria tähän ongelmaan ja kehittäneet mallisovelluksen, joka osoittaa, miten luodut ennustemallit voidaan liittää RapidBase keskusmuistipohjaisen aktiivisen tietokannan yhteyteen ja miten hermoverkkomalleja voidaan käyttää tosiaikaisesti tulevien arvojen ennustamiseen [Hii00a]. Tässä raportissa jatkamme tätä aiemmin raportoitua analyysityötä tutkimalla laajennettua mittausaikasarjaa ja keskittyen Itseorganisoivan Kartan (SOM) soveltuvuuden selvittämiseen.

Olemme käyttäneet aikaviivästettyä viivelinjaa mittausaikasarjan ajallisten piirteiden upottamiseen näytteisiin ja korrelaatioanalyysiä ja tilastollista riippumattomuustestausta hyödyllisimpien syötemuuttujien tunnistamiseen. Korrelaatioanalyysin mukaan tyypillisesti vain aivan viimeisimmät mittaukset korreloivat tulevien arvojen kanssa.

Epänormaalin käyttäytymisen tunnistamiseksi olemme opettaneet SOM-kartan normaalia prosessitilaa kuvaavalla aineistolla. Opittua karttaa on käytetty epänormaalin käyttäytymisen tunnistamiseen uusista testinäytteistä monitoroimalla näiden näytteiden lyhintä etäisyyttä opituista SOM-kartan koodikirjavektoreista. Optimaalisin luokitusraja on valittu kalibrointijoukkoa käyttämällä. Testitulokset osoittavat, että tyypillisesti vain lyhyen aikavälin ennusteet ovat mahdollisia tutkitun prosessin tapauksessa. Parhaat saavutetut luokitustarkkuudet ovat 80-90% luokkaa.

SOM-mallia käytetään myös ennustamiseen opettamalla se satunnaisnäytteellä koko aineistosta ja indeksoimalla kuhunkin SOM neuroniin ne opetusnäytteet, jotka sopivat parhaiten juuri tähän kartan neuroniin. Tätä indeksointia käytetään lähimmän naapurin menetelmään perustuvaan ennustamiseen, kun monitoroidaan uusia näytteitä. Testitulokset osoittavat, että lyhyillä ennusteaikaväleillä ennusteet ovat lupaavia, mutta tarkkuus heikkenee nopeasti pidemmillä ennusteaikaväleillä. Saavutetut luokitustarkkuudet olivat hieman parempia kuin edellisellä menetelmällä, mutta tällä menetelmällä ei pystytty kontrolloimaan väärien hälytysten tiheyttä.

Table of Contents

1 Introduction

We consider the problem of predicting abnormal situations in advance so that corrective actions can be taken in time. The prediction task is considered more thoroughly in the case of an activated sludge waste water cleaning system. In the next chapter we describe alternative approaches to detect abnormalities. Then we describe the industrial process under study.

1.1 Problem formulation

The problem of detecting abnormal situations may be formulated in different manners motivating alternative approaches to the problem:

- Often in process environments normal behaviour clearly dominates the available process data and information of abnormal states is not available. Here one may try to model the normal behaviour of the system by clustering the measurement data records into similar prototype vectors which then describe the typical states of the system. Then one may monitor the distance of each new state from the closest prototype and if the distance is above a given threshold value an abnormal state has been identified and more detailed analysis methods may be started. [Iiva97]

- Another possibility is to cluster the temporal states into typical prototypes using a topology preserving clustering algorithm like SOM and then to label these prototypes according to the portion of abnormal states in the system. Such a SOM map may be used to visualize the trajectory of the system states while monitoring the system and also an alarm may be issued when a critical state is reached. [Alho99]

- Also clustering may be used to divide the system states into typical ones and then to create a local prediction model for future situations based on the examples assigned to the typical states. Based on the amount of examples available the models may range from simple nearest neighbour averaging methods or linear regression models to nonlinear neural network models.

- One approach is to build a classifier assigning each of the process conditions into typical process states based on the historical measurement records available at the time of the classification task. This way the system could warn the user about potential problems before the quality indicators show problems. Examples of good and bad behaviour are needed for building classifiers. The state labels could be normal or abnormal states identified based on the value of the desired quality indicators after the prediction time interval has passed (future values).

- Another approach is to try to predict the continuous-valued outputs of the time series variables a few time steps into the future using time series prediction methods. If the predictions are reliable enough one may build decision support tools on top of such prediction capability for simulating the process and for making "What-If"-scenarios of the process behaviour when some of the values have been changed.

1.2 Description of the activated sludge cleaning process

Our test case concerns two biological subprocesses of an activated sludge waste water cleaning system being used at the Kirkniemi factory of M-Real company. Here we shall briefly describe the relevant aspects of the process studied.

Activated sludge waste water cleaning is a complex process consisting of mechanical, biological and chemical subprocesses. The waste water is continuously fed into the process and then flows through the process for several days. The considered process consists of two partly connected cleaning lines both containing an aeration basin and a sedimentation basin. Before entering these lines the waste water has been mechanically cleaned and its acidity has been neutralized and additional nutritions have been added to the system. The cleaning is based on allowing the bacteria, protozoans and other micro-organisms to eat organic waste as nutrition.

In the aeration basins additional oxygen is dissolved into the water so as to keep the process aerobic and so to allow and accelerate the process. The micro-organisms and the organic waste form a biosludge, which is extracted from the water in the sedimentation phase. A selected part of the sedimented sludge is circulated through the process back to the aeration basin to be reactivated in order to reach a predefined sludge age (meaning the average amount of time that a particle is kept in the circulation). To maintain stability of the process and good quality of the sludge, excess sludge is periodically removed from the process and dried. Removing the sludge too soon may produce too much waste to be dried and to be transported to a dumping place. Also cycling the waste water too long in the system consumes excess cleaning capacity.

The circulation process is slow, a typical delay between beginning and end of the circulation process is 2-3 days. Furthermore, the biological aspects of the process have longer delays, typically 10-15 days. An overview of the considered process is presented in Figure 1-1.

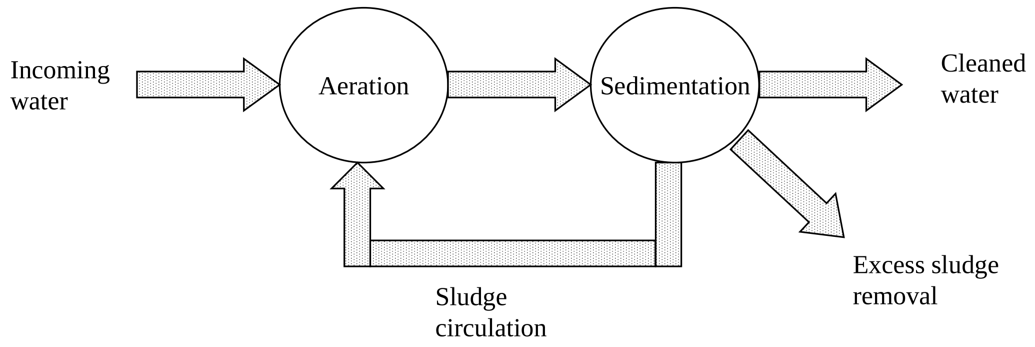

Figure 1-1 Biological waste water cleaning process.

The process operators control the process based on process measurements, such as BOD[1] and COD[2] levels, and prior knowledge. In normal situations, when quantity and quality of the incoming water is stable, control of the process is easy and well known. In abnormal situations, for example, when chemical concentrations of the input water change rapidly, the process can go out of balance. The biology reacts slowly to changes in water quality and quantity, therefore out-of-balance situation in the waste water treatment plant occur several days after the actual cause has occurred. Furthermore, corrective actions are also slow and the results are usually observable only after quite a long delay period.

Process measurements are divided into two basic types: online meterings and laboratory analysis. Online meterings are obtained using sensors and the values from online meterings are available instantly, together with a timestamp. Sometimes online meterings may produce erroneous indications. This is due to fouling of the sensors and drifting of sensor calibration. Typically values from online meterings are available with a time resolution of several seconds, but in this case study averaged values of one hour are used.

Most process quality results are obtained using laboratory analysis. Analysis are performed by taking a sample which is then analyzed in laboratory by chemical experts. Results of the laboratory analysis are entered to the computer system manually with a timestamp characterizing the time when the sample was taken. Most of the laboratory analysis are performed in-house, but some analysis are performed in external neutral laboratories. Typically, results of in-house analysis are available within hours (by late afternoon) after the sample period has ended, but external laboratory results can have a delay of several weeks. The sample period may extend for several days.

1.3 Related research

In [West00] the problem of identifying process conditions in an urban wastewater treatment plant (an activated sludge process) is discussed. Both parametric (Fisher's discriminant analysis, multivariable regression and two-stage regression) and nonparametric (k-nearest neighbours and kernel density) statistical methods are compared with artificial neural network models (Multi Layered Perceptron (MLP) network, Radial Basis Function (RBF) network and Fuzzy ARTMAP) are compared on the problem of classifying a given process state to a predefined set of normal states. Very good results are reported on the ability to discriminate between abnormal and normal situations with RBF and MLP networks and also with Fisher's discriminant analysis. They investigated the false positive rate (Type I error) and false negative rate (Type II error) and with the best models they achived nearly perfect success (97-100%). The ability to discriminate between 13 typical states was about 90% accurate with the best methods. Their data consisted of 527 daily measurements of 38 sensor readings from one urban wastewater treatment plant. They used 10-fold crossvalidation in order to validate the produced models. However, it remained unclear how the different system states were determined - it seems that they were acquired by some type of manual classification. Also it was not

[1] The biological demand for oxygen (BOD) is a measure of the amount of oxygen used by micro-organisms to decompose the organic matter in the wastewater.

[2] The chemical demand for oxygen (COD) is a measure of the amount of oxygen used to oxidize organic matter and to convert it to carbon dioxide and water.

stated what the time span for the forecasts was in the experiments – therefore one would assume that they produced one day forecasts. The authors considered RBF networks as slightly more robust and accurate than the other methods including MLP networks. The other methods were not considered robust enough with the amount of data available.

SOM based clustering and local linear models have been applied in [Vesa97] in predicting a univariate chaotic time series. Their method is based on first embedding the temporal dimension using a time window of historical values as a delay line. Then SOM clustering is applied to identify a representable prototype set for the delay line feature vectors. Next each of the data records is assigned to their BMUs (best matching units) and a local linear model is calculated for each prototype using the assigned data records or by expanding the context by considering the data records of the topologically neighbouring units as well. Also more complex local models, like MLP networks or splines, could be developed if the amount of local data records is large enough. Good results were reported for the selected chaotic time series test case.

In [Kosk97] a similar approach is described but in this case a temporal version of SOM, called Recurrect SOM (RSOM), was used for the clustering and linear regression for the local models. The method was applied to three test cases including a Mackey-Glass chaotic time series test problem, a laser time series and electricity time series forecasting. The method was compared to a MLP network solution and to AR model. The test results indicate that in this case the RSOM method was not as successful as ordinary SOM in [Vesa97] and MLP was typically the best performing method is these test cases and AR only better than RSOM with the electricity forecasting case. These problems may be due to implementation problems in the RSOM algorithm.

In [Simu99] various ways of using the Self Organizing map (SOM) technology are described. It is indicated that SOM maps may be used to visualize the operating point (the BMU of the current measurement) or the trajectory (formed as the set of BMUs of the latest measurements). If the SOM map has been created using all measurement data the map units may be labeled according to the abnormality ratio of the measurements whose BMU it is. In this way the visualisation will indicate if one gets to an abnormal state (assuming that some of the prototypes show a clear abnormality level).

In order to detect abnormalities in the process behaviour (fault detection) one may teach the SOM using only measurement vectors describing the normal behaviour therefore creating a mapping of the "normal operation". A faulty situation may be detected by monitoring the quantization error (distance between the input vector and the BMU). Large errors indicate that the process behaviour is different from the normal data that was used for training the map. By defining a suitable threshold we can select the sensitivity for the failure (novelty) detector.

2 Outline of the performed analysis work

In this case study real process data was used as the basis for building the models. It consisted of operational data stored in history databases during the normal system operation. No planned tests were performed with the controls of the system.

It was decided to gather relevant online measurements and results from laboratory experiments for two partly connected aeration lines and the task was set to develop models for predicting undesirable, abnormal conditions in the target variables. The required raw measurement data was specified together with the customer and it was extracted from the factory process database occasionally and delivered to analysis. The data was combined into a multivariable time series. The target variables include the following 3 indicators: LiukP3 (dissolved phosphorus in the cleaned waste water), CODOUT3 (chemical oxygen comsumption of the cleaned waste water) and JSSAK3 (thickness of the cleaned waste water).

In our previous studies we have described analysis work performed on similar but shorter data set. We previously developed time series prediction methods based on Multi-Layered Perceptron (MLP) neural network models [Hii00a] and Bayesian belief network [Hii00b] models for predicting the future values of the target variables and for classifying the future states into normal and abnormal ones. However, the achieved prediction and classification accuracies were not adequate for applying the results in an operational environment. Therefore, it was decided to try methods based on Self-Organizing Maps (SOM) for abnormality detection.

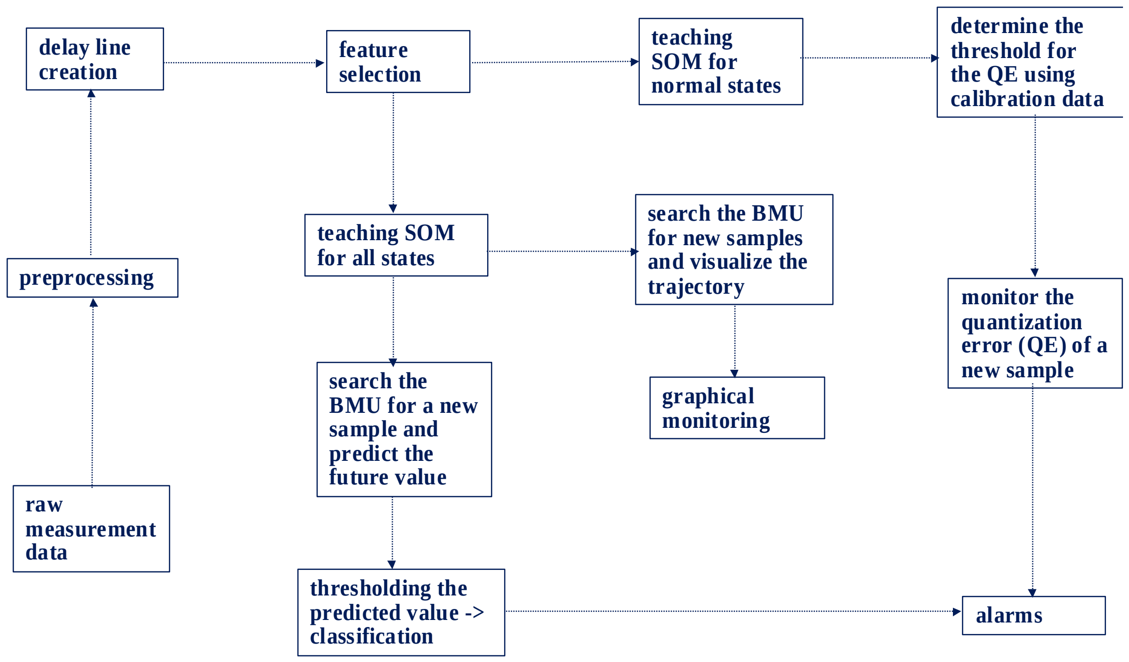
delay line creation
feature selection
teaching SOM for normal states
determine the threshold for the QE using calibration data
preprocessing
teaching SOM for all states
search the BMU for new samples and visualize the trajectory
monitor the quantization error (QE) of a new sample
graphical monitoring
search the BMU for a new sample and predict the future value
raw measurement data
thresholding the predicted value -> classification
alarms

Figure 2-1 The steps taken in the analysis.

The main steps of the analysis work are outlined in the attached diagram in Figure 2-1. At first the raw measurement data is preprocessed as explained in chapter 3. The temporal behaviour is captured by lagging the variables for some days thereby creating a delay line of inputs. The most influential features are selected as explained in chapter 4. Then various modeling tasks are performed with the Self-Organizing Map (SOM). At first a SOM is tought to model the normal operation of the process by using only normal samples as teaching data. Then the level of abnormality is monitored for new samples by computing the quantization error. If this is above a selected threshold (can be objectively estimated by using separate calibration data) the sample is classified as abnormal which may be notified to an operator via an alarm system. In another experiment, a SOM is tought the whole process dynamics by using a sample of the whole data. Then for new data one may determine the best matching unit (BMU) and visualize the traversal of the system state by visualizing the successive BMUs as a trajectory. In another approach an index is formed linking the SOM neurons to the teaching samples that best match that neuron. Then for new samples the BMU can be determined and the target variable mean value over all the teaching samples of that BMU can be used as the predicted value. These are better explained in chapter 5.

3 About the measurement data and its preprocessing

The required raw measurement data was specified together with the customer and it was extracted from the factory process database occasionally and delivered to analysis. The data was combined into a multivariable time series which finally contained 700 daily measurement values starting from 1.4.1999 and lasting until 28.2.2001. The gathered variables had varying sampling periods ranging from hourly measurements to twice a week measurements. It was decided to harmonize the sampling interval to a daily basis. The customer provided the daily values. After the preprocessing 615 samples remained for analysis. In the Ilmas3 aeration line there are five online measurements (such as temperature, water flow, oxygen contents) and eight laboratory measurements (measured 2-5 times per week) and also four descriptive calculated values (available after the laboratory values are available).

Due to the harmonization to a daily sampling period, the harmonized data contained missing values for the measurements performed more seldom. These were estimated by linear interpolation from the data in the preprocessing stage. Suspicious outliers were removed and also these had to be interpolated from the remaining values. Smoothing operations were applied to the time series values in order to reduce the measurement noise.

The smoothing operations were selected for each variable separately and verified with a domain expert. The operations included computing moving averages with different spanning intervals and applying median filtering with different spanning intervals.

Once the measurement values were harmonized into complete time series additional characteristic figures (derived variables) were computed based on the harmonized measurements. The harmonized measurements were divided into two partly overlaping sets each containing the relevant values for the two aeration lines. The preprocessing operations were implemented using the Matlab programming environment.

In Appendix A, we have characterized the preprocessed variables used for Ilmas3 analysis with some statistical figures (mean, standard deviation, minimum and maximum values). In Appendix B, we have included the raw measurements and the corrected values after preprocessing as time series plots.

From these tables and figures it is possible to draw some preliminary conclusions on the problem. The target variables of interest are LiukP3 in Figure B.2, JSsak3 in Figure B.6

and CODout3 in Figure B.8. LiukP3 has frequent peeks rising well above the desired maximum level of 0.5; even the average value is above that level. The peaks typically last a few days but some last longer and also have multiple subpeaks. JSsak3 has few very large peaks (very much above the desired maximum of 50) hiding the smaller variation. This variable should be log-scaled before analysis. CODout3 seems to have very consistent behaviour and the consecutive values do not have very large differences. The desired maximum level of 200 is crossed especially during the start of the analyzed time period.

The time series plots may be used to search for possible causes of the temporal behaviour of the target variables – the cases with too high values are of particular interest. It seems that the shutdowns (VIRTA3 dips) can explain 7 of the 18 major peaks of LiukP3 time series. Actually all the shutdowns cause a sharp rise in LiukP3 level. One of the remaining peaks seems to correlate with a CODin dip around day 330. However, 10 unexplained peaks remain which do not seem to have a clear explanation. In the oxygen level time series (Happi4) there seems to be at least 2 dips (at days 170 and 330) from the usual level which seem to co-occur with some of the unexplained Liuk_P3 peaks. These dips indicate that the oxygen consumption during the aeration has been exceptionally high. On the other hand between days 420 and 480 there are many peaks in Happi4 which co-occur with many unexplained LiukP3 peaks. While examining the flocking load variable (Fkuorma3 in Figure B.9) one notices that the peaks between days 320 and 500 somewhat co-occur with the unstable behaviour of LiukP3.

The nitrogen levels (Typpi3 in Figure B.3) have frequent peaks possibly suggesting that the nutrition is given in bursts.

In Figure B.1, we see that the water flow of Ilmas3 line (VIRTA3) has some clear dips caused by factory shutdowns (typically the process is running continuously but during certain holidays (Mid Summer day, Independence day and Christmas) the factory may be shutdown. Visually one can see that these dips seem to cause changes in (correlate with) some of the variables: the water temperature (Lampo) drops, the waste content in the water (CODin) drops, the levels of oxygen at the start and end of the aeration line in Ilmas3 line (Happi3 and Happi4) rise. It also explains many of the peaks of variable LiukP3 as explained above.

From the above discussion one may notice how difficult it is to find explaining causes for the behaviour of the target variables using the data alone. Also there seem to be many weak influences which manifest differently in different situations due to mutual influences (the explanatory variables are not independent). It also seems that many of the more rare causes are only represented by few instances in the database making there learning difficult.

4 Identification of the most influential explanatory variables

In order to model a large chemical process using neural networks one needs to have a large data set which adequately covers the phenomena to be analyzed or predicted and also has sufficient variation in the parameters. Typically production data is used in such studies as it is often automatically gathered and stored and is therefore readily available. It has been suggested that using experiment design methods could provide a better basis for gathering the required training data. However, this is seldom done due to the time delays and disturbances to the process. We have also used production data. This is also related to the fact that the data needs to contain the desired dependencies as no mathematical method may find one if it does not exist. [Taip99]

The analyzed system is complicated by strong dynamic feedback loops and a large amount of available measurements. However, the behaviour of the process is not well known and the status of the biological process is not possible to measure online.

Irrelevant and redundant variables disturb the teaching of the neural networks. Ideally the inputs should be highly relevant to the task (prediction, regression, classification) and mutually non-dependent. Neural networks are justified by allowing one to capture the nonlinear dependencies in the training data automatically. Dependencies may show up different in different environmental conditions (e.g. when the temperature or its derivative cross a critical threshold). Also the phenomena may have hysteretic behaviour where the process status does not return to the normal state once the environment variables stabilize [Rita98]. The behaviour of the waste water cleaning process seems to have such a phenomena when the status of the biological cleaning process gets disturbed.

In order to support modelling one needs to concentrate on the most relevant explanatory variables by selecting the best subset of the variables. Ideally this could be done by testing different combinations of the input variables by generating the desired models on that information and then evaluating the performance based on an independent validation data set. A genetic algorithm based stochastic search algorithm could guide the search for the best models. This was tested with our previous tests using the NGO (NeuroGenetic Optimizer) tool. However, it is a computationally heavy approach and gives little knowledge on the cause of the suggested dependencies. Using simple correlation analysis one may detect linear dependencies between a couple of variables. At least these should be included into the model but nonlinear dependencies remain undetected. Also detecting the redundant variables is difficult. Statistical dependency testing (using contingency tables) may reveal more dependencies than the correlations. Also entropy based mutual information measures could be used to select a proper variable set.

Correlation tests are suitable for detecting linear dependencies between random variables. Nonlinear dependencies may be missed. Anyhow, one should at least include those variables found by correlation analysis.

Statistical dependency testing may be used when there are enough samples. However, in our case we found that they only allowed us to eliminate the most clearly independant connections. Therefore more pruning would be needed.

4.1 Correlation analysis

We have created a small utility program for searching correlations between the desired variables and their previous values. At first the original data set is expanded by adding new columns for each variable corresponding to its delayed values (1-to-N steps). Then the correlation matrix is formed and the desired dependencies are plotted on a figure with a grey-scale format on the left and a thresholded (>= given threshold) one on the right (see Figure C.1 in Appendix C). Also the dependencies that have a larger correlation as the threshold are reported in a textual format (see Listing C.1 in Appendix C).

The report in Listing C.1 reports the correlation based dependencies of the target variables (values 3 days into the future) with a maximum delay of 21 days. One can see that with the moderate threshold (0.5) LiukP3_F3 is only dependent on the current value of LiukP3 and three past values of the flow variable (Virta3). The same matters can be seen in Figure C.1 where the first row in the figures shows the dependencies for LiukP3_F3, the second for JSsak3_F3 and the third one for CODout3_F3.

In Figure C.2 and the listing C.2 we have also shown the dependencies for the whole data set with a larger threshold (0.7) and a maximum delay of 21 days. From the left side plot (in Figure C.2) we can see that there are temporal and intervariable dependencies in the data but few are larger than the selected threshold 0.7. This is also confirmed by listing C.2.

4.2 Statistical dependency detection using contingency tables

In order to be more conservative in eliminating explanatory variables we evaluated statistical dependencies between the potential pairs of explanatory and target variables. The delayed data columns were first discretized into bins with equal amounts of samples. Then a binned table (the contingency table) was formed for each pair of potential variables (the explanatory candidate and the target variable) and it was tested with the CHI2 statistical test how big the risk of these variables not being independent is (i.e. it was tested whether P(A,B) == P(A) * P(B)).

In Figure D.1 in Appendix D we have plotted the probability of the pairwise dependencies in the left plot and those probabilities being above the 0.1% risk level in the right plot. By viewing the Listing D.1 we see that only few of the possible dependencies could be pruned using this approach.

5 Using SOM to analyze the waste water data set

We have applied the Self-Organizing Map (SOM) neural network method to the waste water data in two different ways: First we have tested how well a SOM, that has been tought with normal data, can be used to detect abnormalities in previously unseen testing data. Next we have tested how a SOM, that has been tought with a random sample of all the gathered data, can be used to predict the response on new unseen samples. We have run the SOM tests in the Matlab mathematical programming environment using the SOM toolbox [Vesa00]. The basic principles of the SOM algorithm has already been well described in another Modus-project report [Rint00].

5.1 Abnormality detection using SOM

Abnormality detection or novelty detection using SOM is based on the idea of approximating the probability distribution of the data samples representing the normal system behaviour. The prototype vectors in the trained SOM neurons represent these approximations compactly. The degree of novelty of new samples may be monitored by measuring their distance (usually Euclidean) to the closest unit (BMU, the Best Matching Unit). This distance is also called the quantization error. For normal samples, this quantization error is typically smaller than for the abnormal ones which enables one to build a classifier to predict novel states in advance. Similar approaches have been described in [Alan91], [Alho99] and [Simu99]. The justification for using this approach is that when modeling real processes normal data is typically much more common than abnormal failure data. Therefore a good approach to detect failures is to monitor any differences from the normal behaviour and then examine such cases more carefully with some other means.

In our test environment the sample data was first enriched by embedding some of the past input values into the feature vectors, thus creating a delay line of inputs. A maximum delay interval of 21 days was used. Next the most influential features were identified using correlation analysis tests and the selected ones were included into the final feature vector. The normal data was selected based on certain process conditions being normal (the incoming water flow VIRTA3 being above a critical level, $>= \mu - 2 * \sigma$) and the target variable being below a critical level supplied by a process expert. The data samples were then divided into different sets of data: only normal data (60% of the normal data samples) were assigned into the learning data set. A mixture of normal (25%) and abnormal data (60%) was assigned into the testing data set. A mixture of normal (15%) and abnormal data (40%) was assigned into the calibration data set. The teaching data was used to teach the SOM mapping.

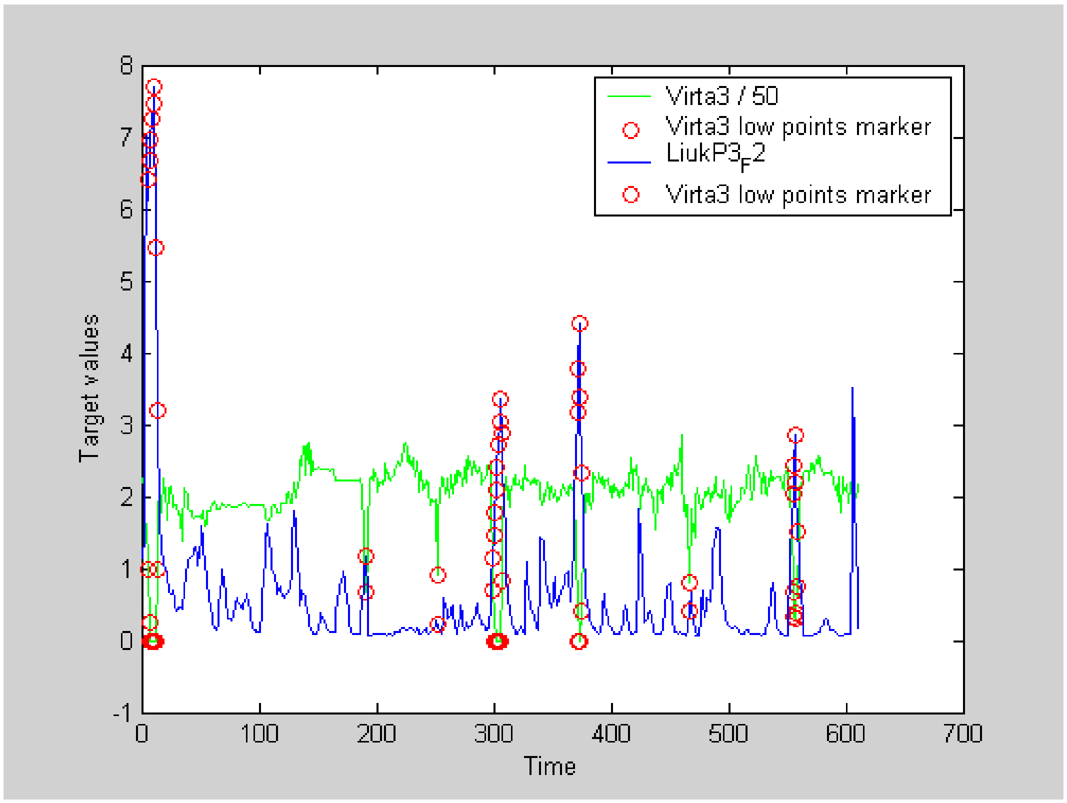

Figure 5-1 The effect of process shutdowns (Virta3 with the green line) on the target variable LiukP3 levels (the blue line).

In Figures 5-1, 5-2 and 5-3, we show the effect of process shutdowns on the target variable values. In Figure 5-1, we can see that low values of Virta3 co-occur with most of the LiukP3 largest peaks therefore explaining these cases. In some occasions the peak is not very high if the shutdown has lasted for only a short while. This clearly indicates that the shutdown data should be handled differently. Predicting their effects gives no information as they are prescheduled and planned.

From Figure 5-2, we can see that the process shutdowns do not have an immediate effect on the target variable CODout3. However, it seems that there may be a delayed rise in this target variable a few days after the incident.

From Figure 5-3, we can see that the process shutdowns do not have an immediate effect on the target variable JS_sak3 and at least the large values are not explained by this incident.

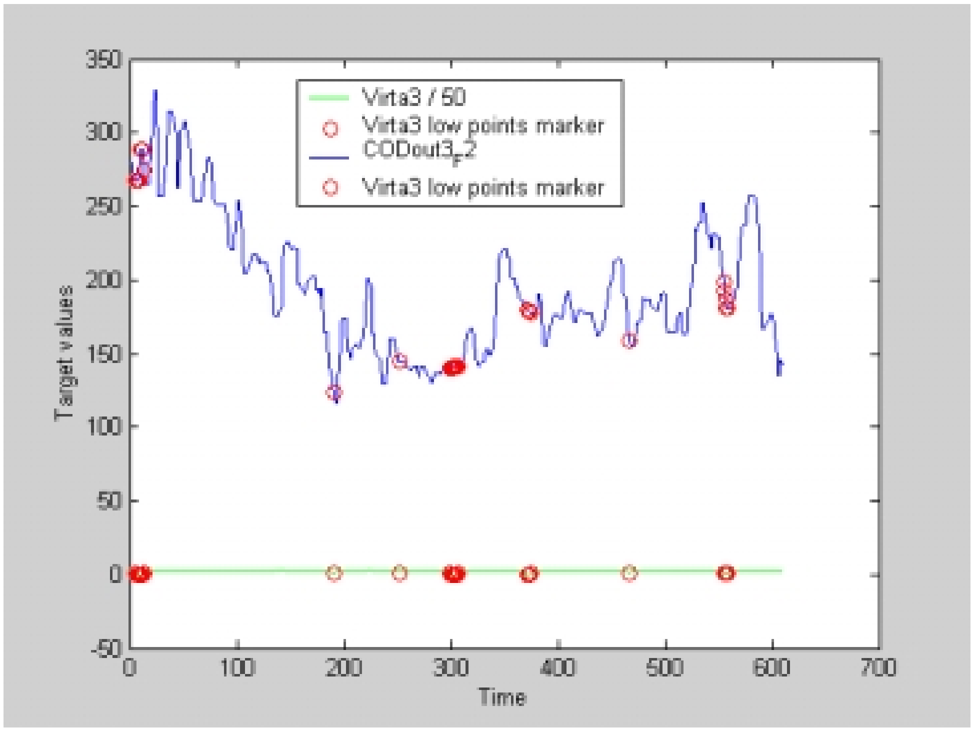

Figure 5-2 The effect of process shutdowns (Virta3 with the green line) on the target variable CODout3 levels (the blue line)..

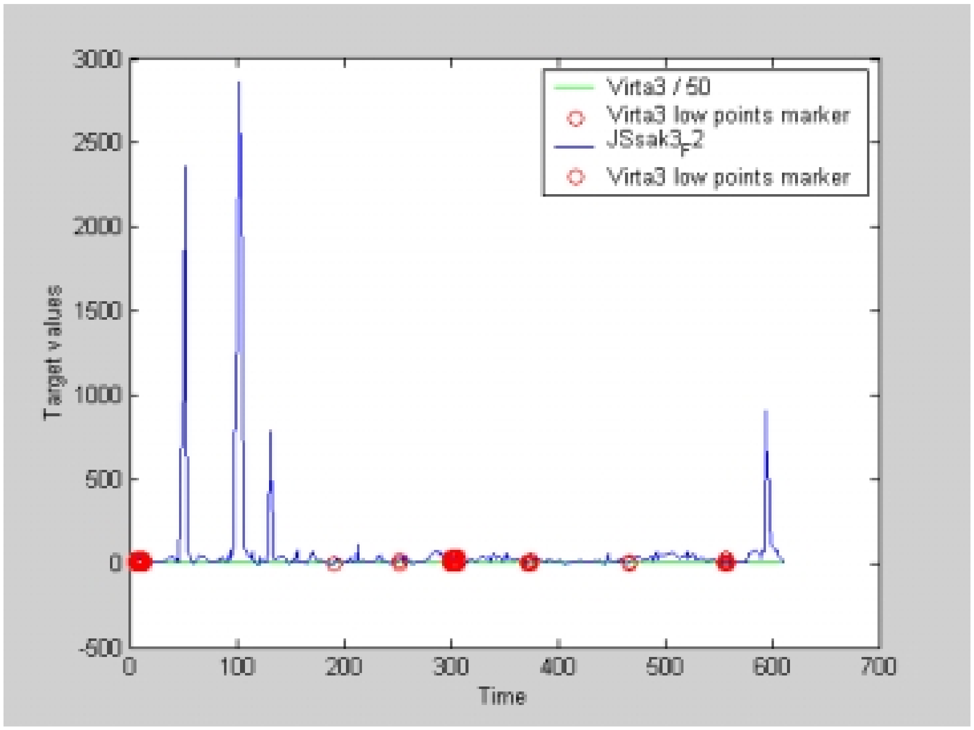

Figure 5-3 The effect of process shutdowns (Virta3 with the green line) on the target variable JSsak3 levels (the blue line).

The SOM was used to detect the abnormality in the testing data by monitoring the quantization error (QE) to the best matching unit (BMU, the closest unit). As the quantization errors for normal and abnormal samples are partly overlapping (see Figure 5-4) a threshold value has to be defined to optimally divide the new samples into normal and abnormal ones. Calibration data was used to select the threshold value for the quantization error by searching for the threshold value producing the best classification accuracy. Alternatively, a more advanced cost function for misclassification costs could be optimized by assigning different misclassification costs to the different classification errors. In Figure 5-5, we can see an example of the compromise between classification accuracy and the false positive and the false negative rates. We can see that as the threshold value increases (one is classifying samples to an abnormal category if the QE value is larger than the threshold) the rate of false negatives increases and the rate of false positives decreases. The optimal classification accuracy is typically achieved when the false positive and false negative rates are approximately equal. [Moze00] provides further advice on choosing the appropriate threshold value for classification purposes.

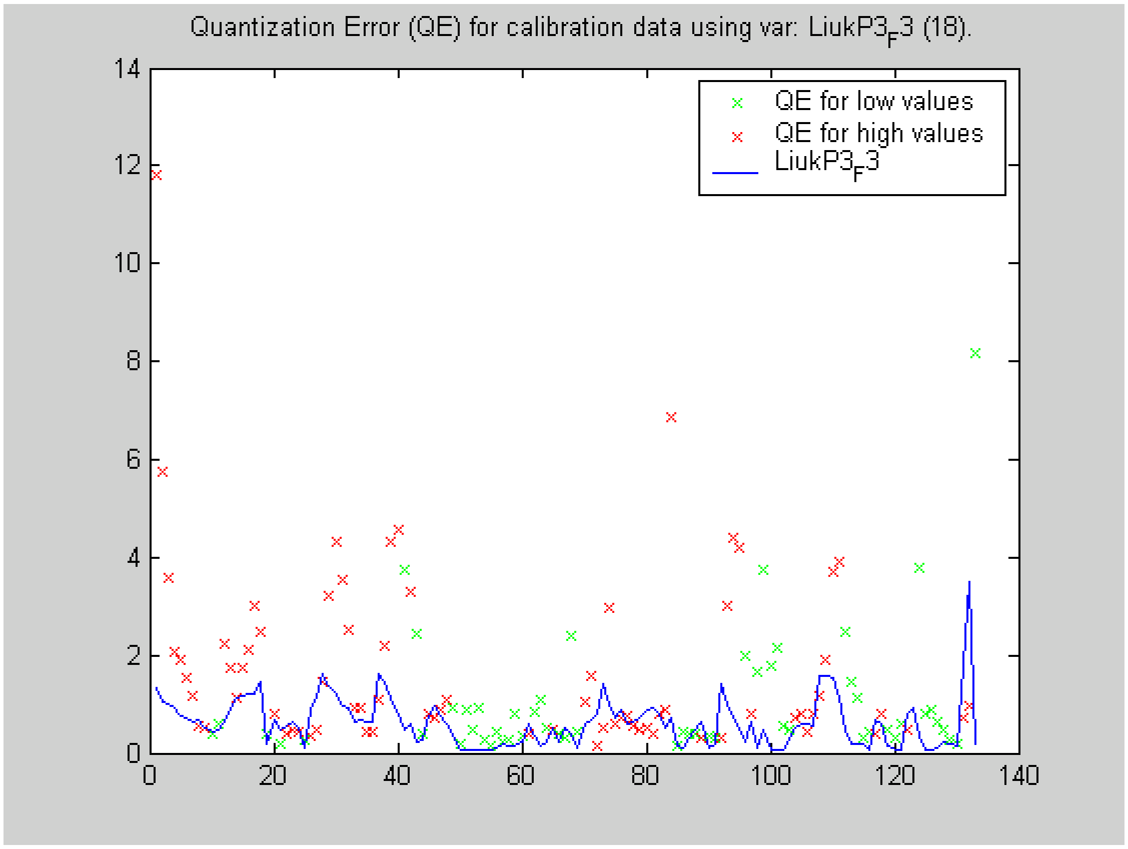

Figure 5-4 An example of the Quantization Errors for a mixture of normal and abnormal data comprising the calibration data set that is independent of the teaching data set. The crosses indicate the QE-values (red ones mark the QE-values for abnormal states and the green ones the QE-values for normal states). The blue line plots the target values. We can see that in this case the QE-values for the different states are highly overlapping thereby preventing perfect classification.

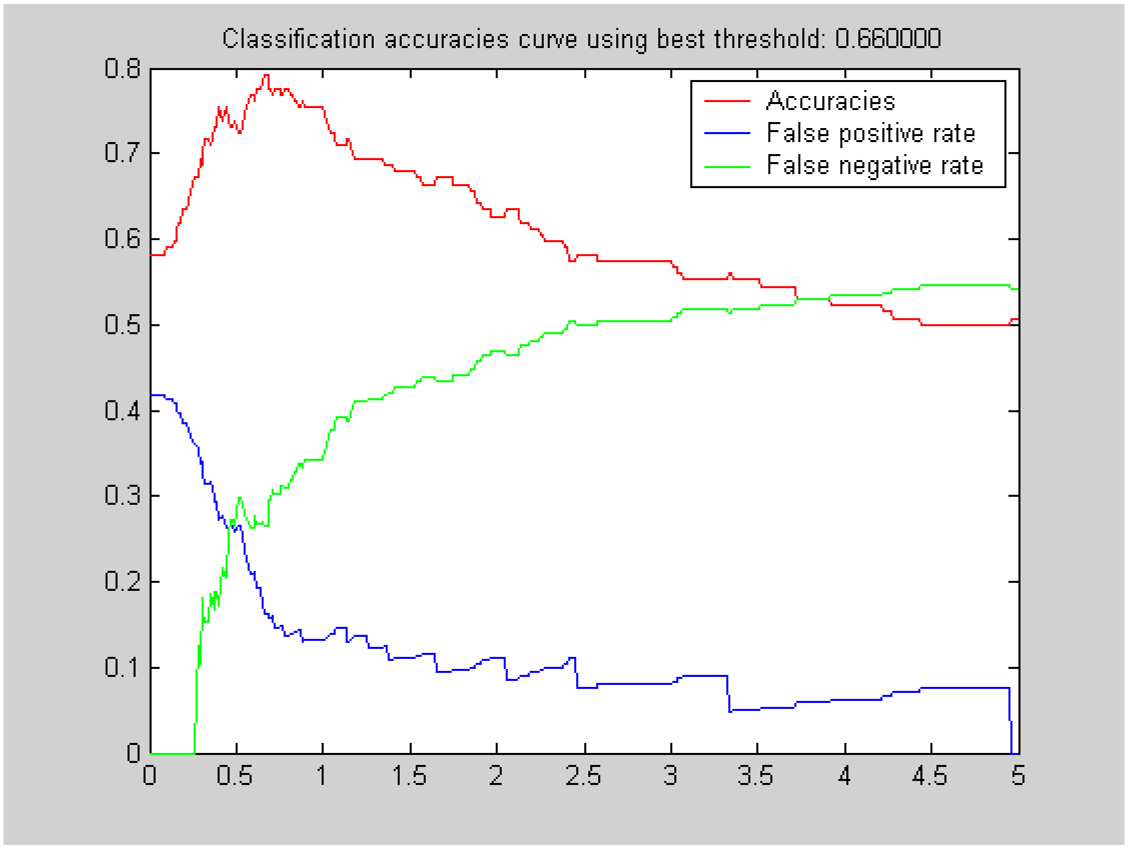

Figure 5-5 An example of the effect of the threshold value on quantization error (QE) to the classification accuracy and the false positive and false negative rates.

In Table 5.1, we see that for short prediction intervals (1 or 2 days) the prediction accuracy is fairly good for variable LiukP3. But with longer periods the accuracy quickly deteriorates. (The 50 % level is equal to random guessing performance). Figure xx.3 shows an example of the performance of the predictions on testing data with 70.9% accuracy. We see that there are quite a lot of errors all over the time series.

	F1	F2	F2	F3	F5
min correlation	0.6	0.6	0.3	0.3	0.3
# inputs	6	4	31	21	7
accuracy	83.2%	73.2%	70.9%	55.7%	53.1%
FP	20.3%	19.4%	29.3%	44.2%	44.9%
FN	10.5%	33.0%	28.8%	45.8%	54.4%

Table 5.1: Accuracy of the SOM QE-based classifiers for different prediction tasks of variable LiukP3 (1, 2, 3 and 5 days ahead) (FP = false positive rate, FN = false negative rate).

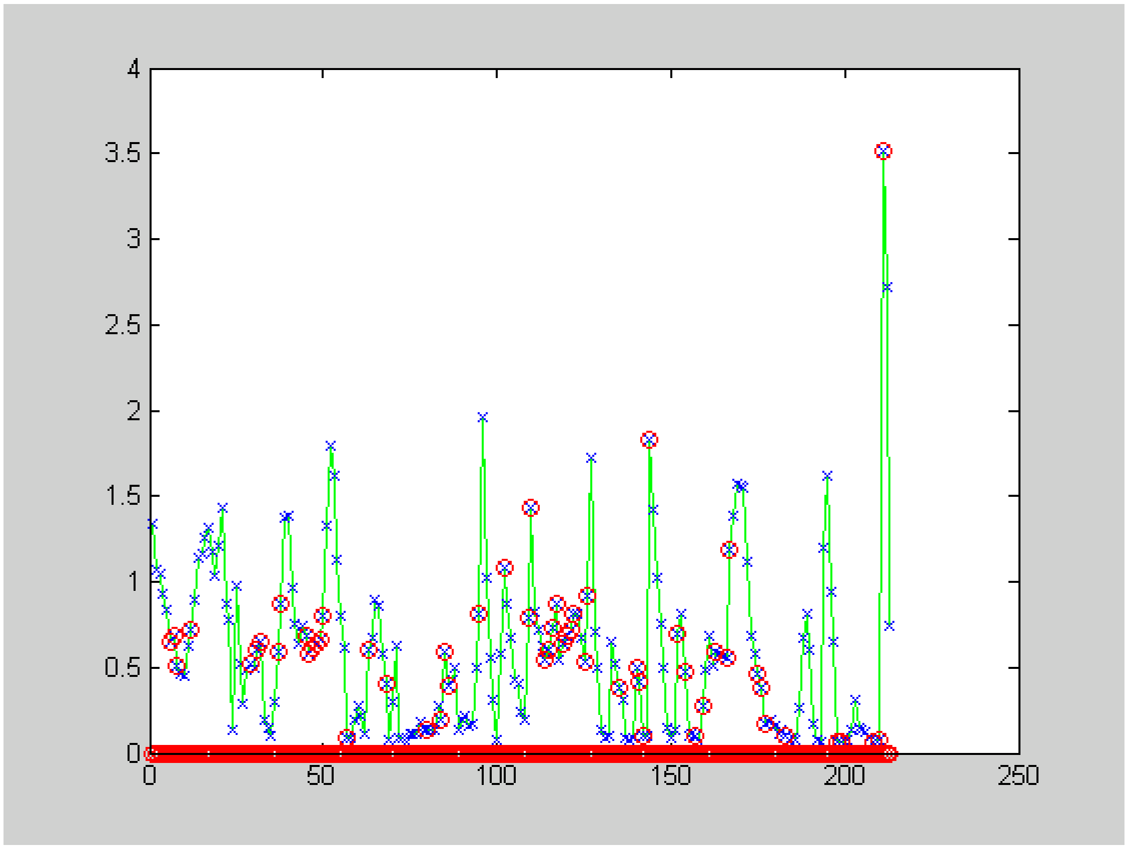

Figure 5-6 A plot showing predictions two days ahead in time for the testing data of time series LiukP3_F2. The prediction task is to try to predict whether the future value (2 days ahead) is going to be high (> 0.5) or low (<= 0.5). The crosses represent real target values. The circles mark cases where the predictive classification has failed.

In Table 5.2, we see that for short prediction intervals (1 day) the prediction accuracy is good for variable COD_out3. Even with longer periods (10 days) the accuracy is fairly good. Figure 5-7 shows an example of the performance of the predictions on testing data with 1 day prediction. We see that there are some errors especially when rapid changes occur in the target values.

	F1	F1	F5	F10
min correlation	0.6	0.7	0.7	0.7
# inputs	32	22	18	9
accuracy	87.3%	90.1%	74,6%	70.8%
FP	7.4%	7.0%	15.4%	29.2%
FN	19.2%	14.0%	34.8%	29.0%

Table 5.2: Accuracy of the SOM QE-based classifiers for different prediction tasks of variable COD_out3 (1, and 10 days ahead) (FP = false positive rate, FN = false negative rate).

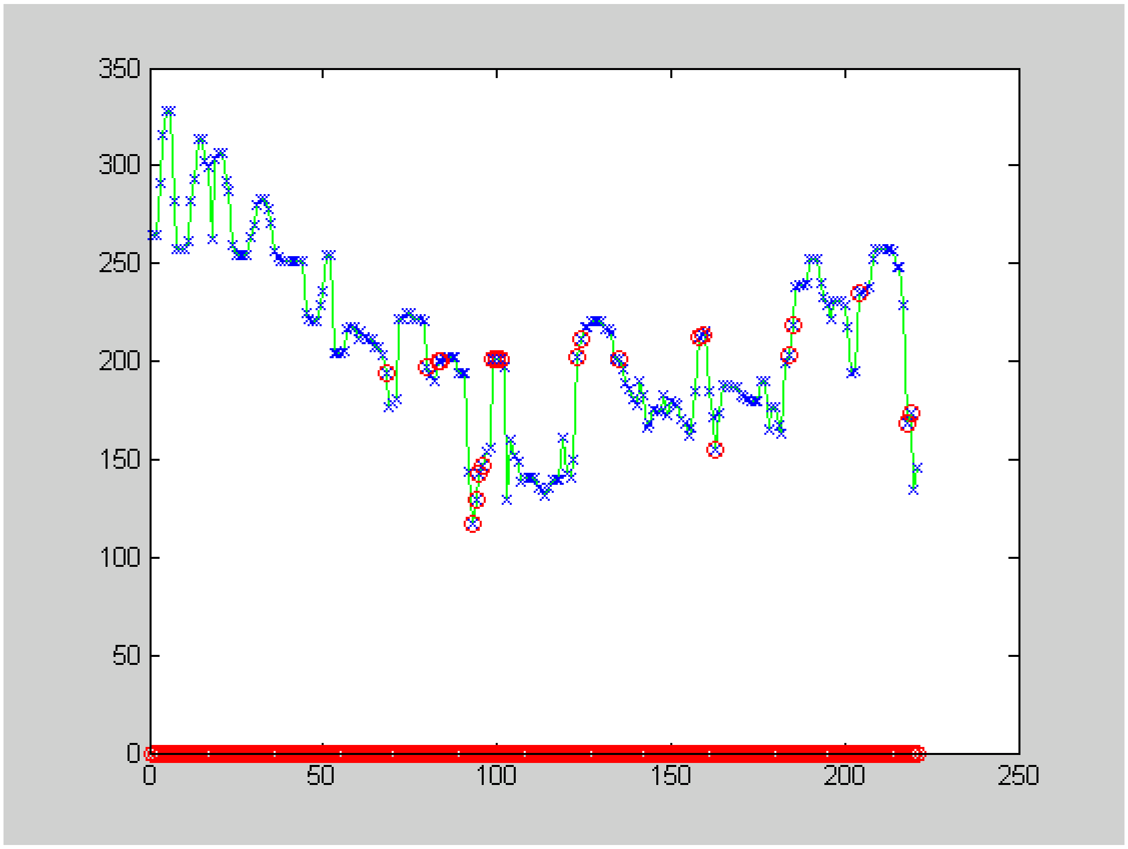

Figure 5-7 A plot showing predictions two days ahead in time for the testing data of time series CODout3. The prediction task is to try to predict whether the future value (1 day ahead) is going to be high (> 200) or low (<= 200). The crosses represent real target values. The circles mark cases where the predictive classification has failed.

In Table 5.3, we see that for short prediction intervals (1 days) the prediction accuracy is good for variable JSsak3. But for longer periods (3 days) the accuracy quickly deteriorates. We can see that the values for this variable only contain very few explanatory variables (in this case only the two previous values of the target variable) which makes prediction very hard.

	F1	F3
min correlation	0.6	0.4
# inputs	2	1
accuracy	88.0%	72.6%
FP	19.2%	39.1%
FN	8.9%	25.7%

Table 5.3: Accuracy of the SOM QE-based classifiers for different prediction tasks of variable JSsak3 (1, and 3 days ahead) (FP = false positive rate, FN = false negative rate).

To summarize the classification performance using this method is not very reliable. The performance is better especially with the CODout3 variables partly because the adjacent values tend to change slowly. With variable JSsak3 the classification performance is high but it seems that most of the test values have been low ones and the false positive rate is very bad. Also only few explanatory variables were found for this variable. It only tended to self-correlate with its previous values. There were quite a few explanatory variables for LiukP3 levels (such as CODin, VIRTA3, LiukP3) but the long term prediction did not succeed well. Only short term prediction was feasible.

5.2 Using SOM to model the overall process behaviour

In another test we have tought SOM with a random sample of all the available test data. In this case the prototype vectors of the SOM neurons specify a compacted vector quantization of the sample space. Next, each of the teaching samples have been assigned to the best matching SOM neurons (BMUs) so that each such neuron has a list of the samples best matching it. These lists may be used for assigning labels to the units indicating the number of samples with different categories assigned to them. The frequencies of such labels may be used as classification probabilities for new data samples. Therefore, one may classify new samples to high and low values based on the portion of the categories in the training set. Alternatively one may build classification or prediction models locally based on the set of learning samples assigned to each neuron. The prototypes may also be grouped together. This allows one to assign enough samples to the cluster nodes in order to build more accurate models. Such approach was used for time series prediction in [Vesa97].

In our testing environment we have assigned a random sample (60%) of the available and time-lagged data set to training and the rest to testing purposes. The data was clustered using a 2-dimensional SOM map. The BMUs were determined for each of the training samples and they were indexed so that for each SOM neuron we had a list of the training samples whose BMU the neuron was.

	F1	F2	F3	F5
min correlation	0.5	0.5	0.5	0.3
# inputs	13	9	5	7
thresholded accuracy	84.0%	82.7%	69.1%	63.0%
FP	25.0%	27.1%	49.4%	49.4%
FN	11.6%	11.6%	19.6%	29.1%
majority vote accuracy	51.5	63.7%	48.7	60.9
FP	71.3	83.3%	68.9	53.3
FN	34.7	32.7%	31.5	34.1
RMS error	0.31	0.38	0.57	0.73

Table 5.4: Accuracy of the SOM based predictions for different prediction intervals for variable LiukP3 (1, 2, 3 and 5 days ahead) (FP = false positive rate, FN = false negative rate, RMS error = Root Mean Squared error).

These indices were used to produce predictions for testing data samples. In order to produce regression estimates of the target variables the best matching unit was determined for each test sample and then the mean value of the target variable among the training samples assigned to this neuron was used as the estimate. For classification predictions the regression estimate was thresholded in order to get the category prediction. Alternatively the category was determined by performing a majority vote among the training samples of the BMU neuron.

In Tables 5.4, 5.5 and 5.6, we have gathered some model performance metrics for various tasks. In Table 5.4, we see that the performance in predicting the abnormally high values is fairly good for short term predictions but deteriorates fast with longer term predictions. Also the false positive rates are fairly high in all the predictions thereby causing false alarms.

In Figure 5-8, we have plotted the classification accuracy for a mean value prediction using threshold classification approach. We can see that still quite many misclassifications occur especially near the maximum allowed normal value.

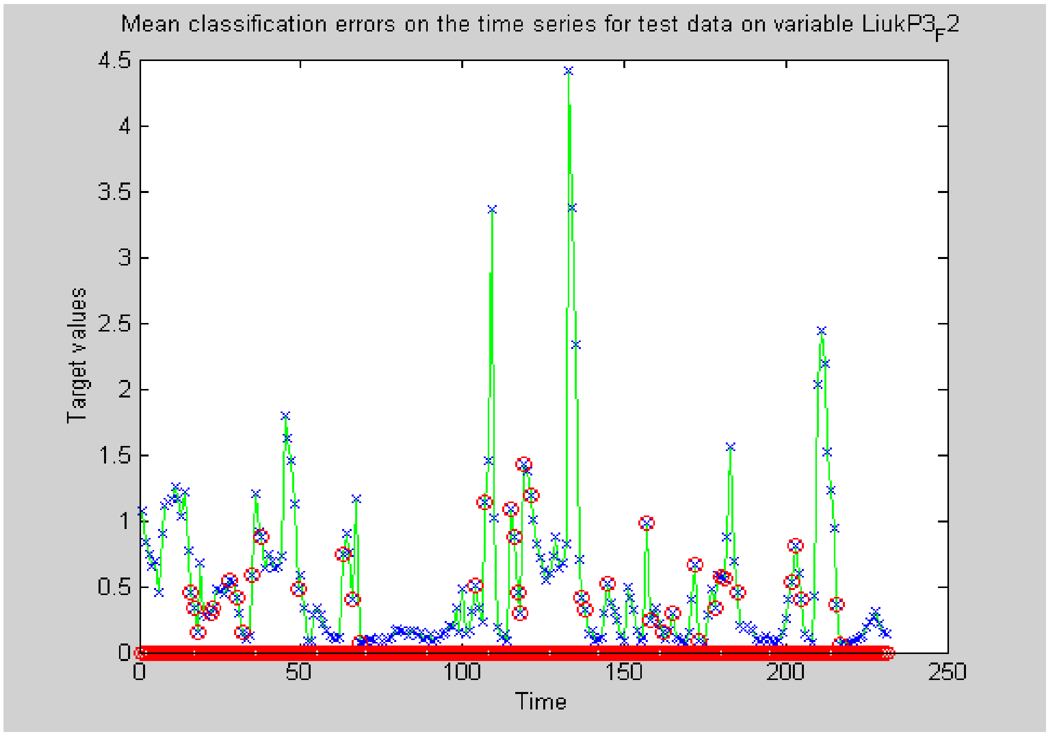

Figure 5-8. An example of the classification performance for variable LiukP3_F2. The blue crosses and the green line mark the desired values and the read circles indicate that the thresholded prediction has been erroneous in that case.

However, the regression predictions seem to work fairly well with the same data. In Figure 5-9, we have plotted an example of the values produced by the mean value predictions for unseen test data. We can see that with this variable the regression predictions follow the desired values quite well. Although, in some cases the predictions lag behind by even 2 or more days, even a lag of 1 day gives some predictive value in this case.

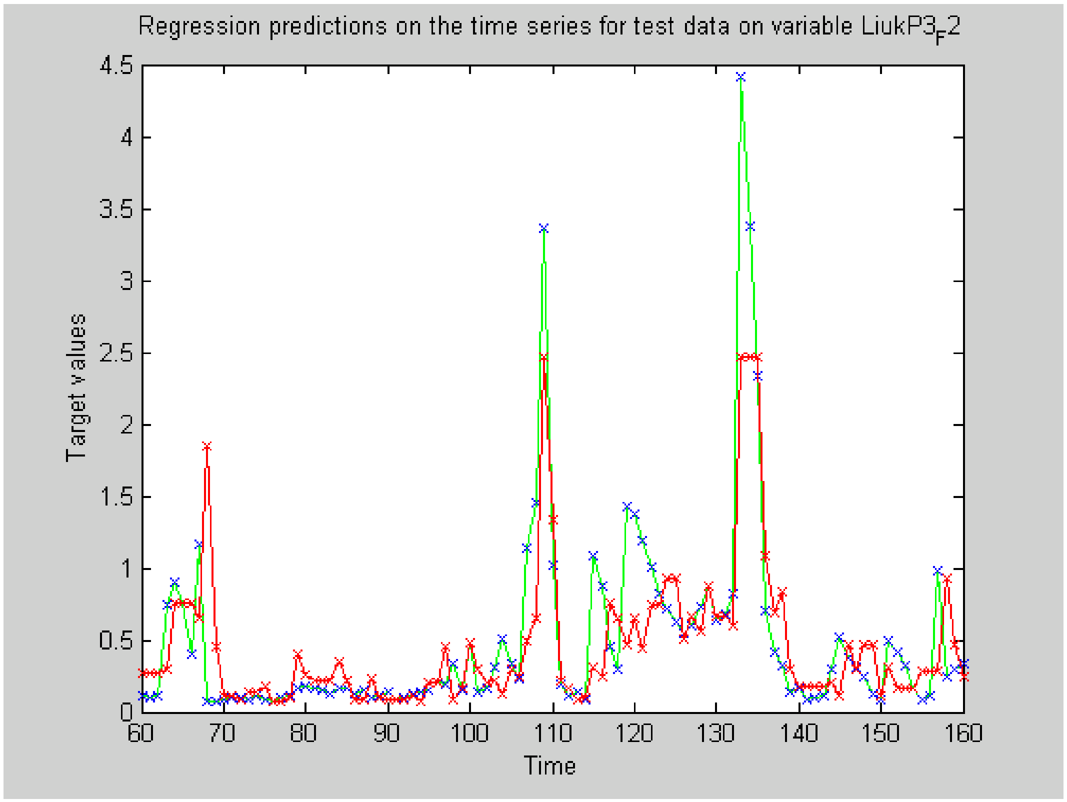

Figure 5-9 An example of the mean value predictions for LiukP3_F2 prediction.

	F3	F7
min correlation	0.8	0.8
# inputs	7	3
thresholded accuracy	85.2%	74.7%
FP	7.4%	38.9%
FN	17.9%	16.6%
majority vote accuracy	48.3%	66.8%
FP	75.4%	41.8%
FN	44.6%	21.9%
RMS error	19.0	30.2

Table 5.5: Accuracy of the SOM based predictions for different prediction intervals for variable CODout3 (3 and 7 days ahead) (FP = false positive rate, FN = false negative rate, RMS error = Root Mean Squared error).

In Table 5.5, we see that while using the thresholded classification on the mean value predictions the performance in predicting the abnormally high values of CODout3 is very good with 3 day predictions and fair even with 7 day predictions.

In Table 5.6, we see that predictions for JSsak3_F3 are poor because of the very large false positive rate. Also only autoregressive dependence has been found by the feature selector. It seems that these methods are not applicable to this task.

	F3
min correlation	0.5
# inputs	1
thresholded accuracy	80.8%
FP	73.1%
FN	12.2%
majority vote accuracy	86.1%
FP	50.0%
FN	10.5%
RMS error	282

Table 5.6: Accuracy of the SOM based predictions for 3-day prediction interval for variable JSsak3 (FP = false positive rate, FN = false negative rate, RMS error = Root Mean Squared error).

In general, it seems that the majority voting based approach does not work well. Classification based on thresholding the regression results works fairly well with short prediction intervals – however the high false positive rates indicate that the models perform worse than the QE-monitoring methods in identifying abnormal situations reliably.

Another approach to use the SOM map is to visualize the state of the process by drawing a trajectory of the BMUs for the monitored samples on top of the map grid. In Figure xx1, we have plotted the 2-dimensional SOM map which was tought to predict LiukP3_F2 values. For each attribute (variable) in the feature vector the attribute value of the appropriate neuron is plotted on the map grid (component planes). By looking for similarities between the component planes one may identify dependencies in the data. For example, the highest values of LiukP3_F3 seem to correlate with the process shutdowns (Virta3 variables having the lowest values). Also the variables Lkuorma3 and CODin have there low values in the same areas and correlate with the other variables as well. The U-matrix shows graphically the Euclidean distances between the neighbouring SOM neurons. In some situations clear clusters are formed in this view by rims of high distances separating clusters with low mutual distances. In this case the only clear rim of high distances is on the left lower corner where the water flow has very low values. All the other values seem to be very similar.

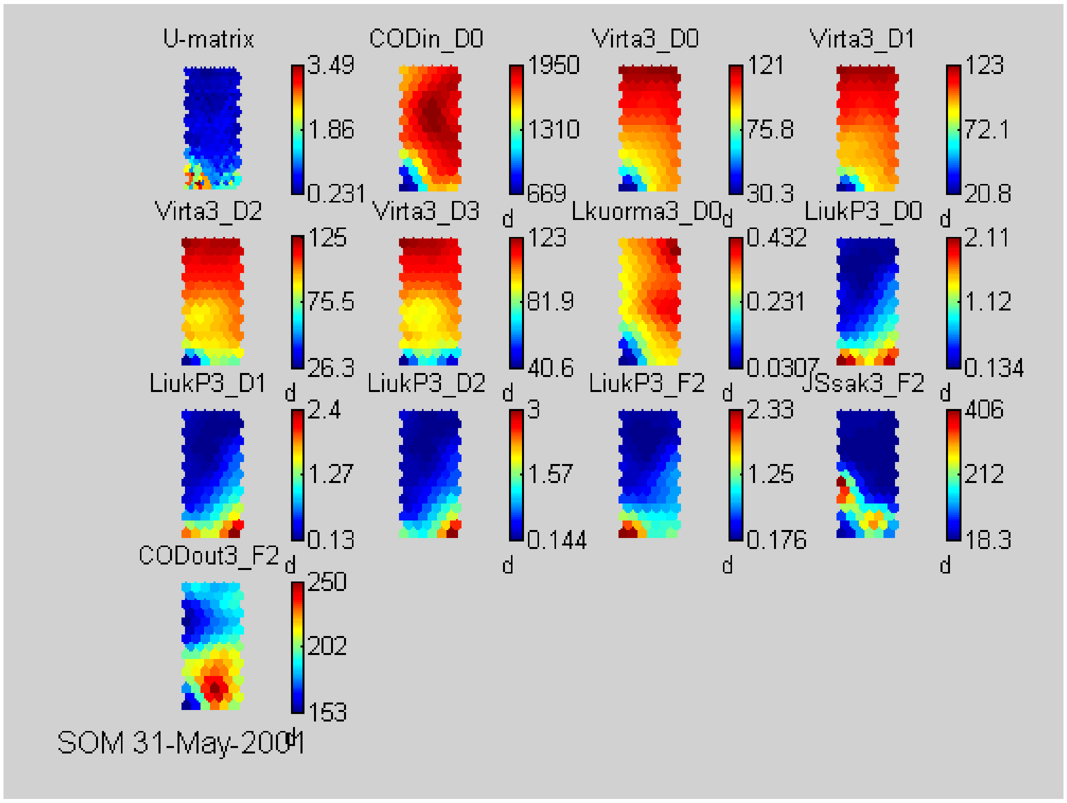

Figure 5-10 A component planes plot of a SOM map created to predict LiukP3_F2 values.

In Figure 5-11, we have plotted predictions for the last 20 values of the available data (validation data in this case). We can see that in this case the predictions seem to be too high for normal situations and fail to predict the last peak. However, this peak was caused by a shutdown period of the factory according to the Virta3 values which makes its forecasting impossible given the time frame.

In Figure 5-12, we have plotted the trajectory of the best matching units (BMUs) for the same calibration data set. The numbers on the neurons indicate the time steps during which the process status best matched that neuron. The idea is that in suitable conditions the SOM map could be used as a monitoring system that shows the system status on the SOM map visually. Some of the map areas would be marked as undesirable alarming states (like the left lower corner as indicated by the high value area of the LiukP3_F2 variable in Figure 5-10). We can see that while the values are normal the trajectory stays in the neutral area and when the values start to rise it moves toward the alarming area and again leaves it when the situation stabilizes. Possibly a better example period would have produced a better proof.

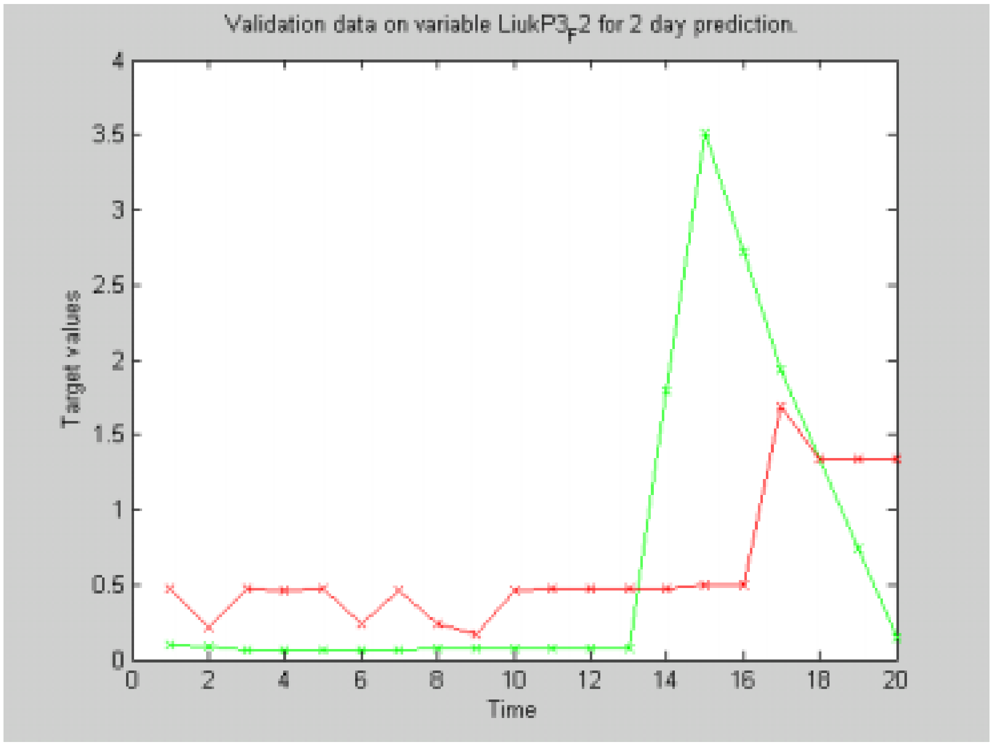

Figure 5-11 A time series plot of the real values for the LiukP3_F2 variable for 20 days. Green values indicate the desired values and the red ones the predictions.

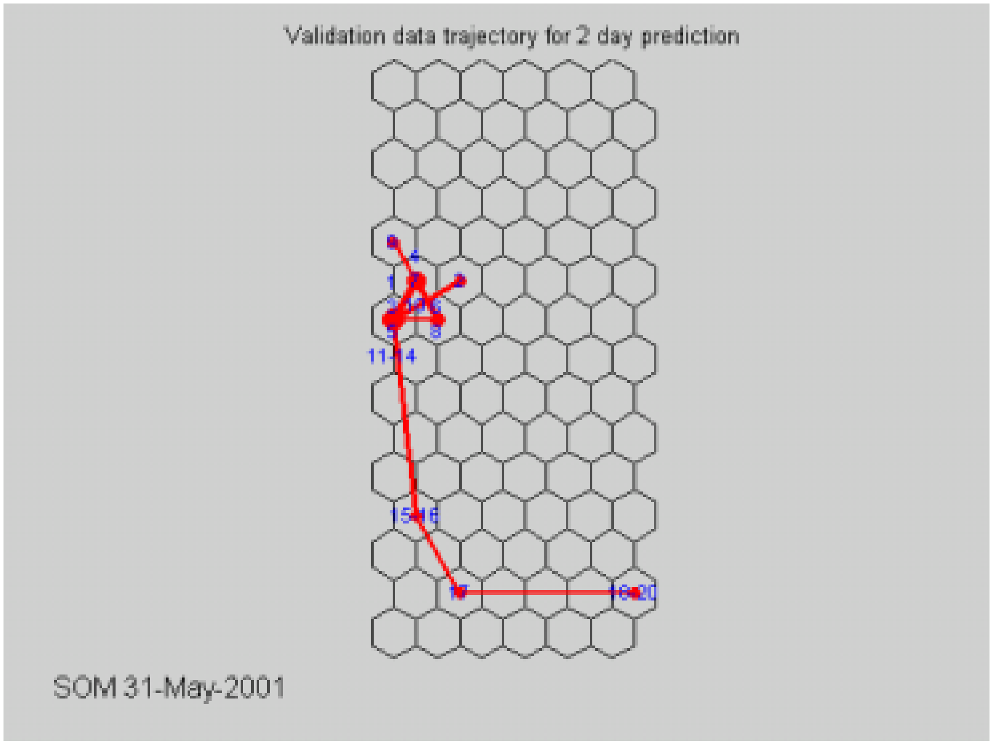

Figure 5-12 Validation data trajectory of predicted for 20 daily measurements of LiukP3_F2. The text labels indicate the days when the process was in this state (neuron).

6 Possibilities to operationalize the generated models

As none of the generated models produced perfect prediction results for new data one may need to combine different models in order to get the best performance. In Figure 6.1 we have outlined some of the possibilities. A calendar based filtering system is used to prune predictions for periods of known abnormal operation (planned production shutdowns should be eliminated as their impact on the target variables is faster than the prediction interval). For identifying the abnormal conditions one might use both the normal state SOM and the overall SOM models and calculate a certainty on the predictions based on their consistency.

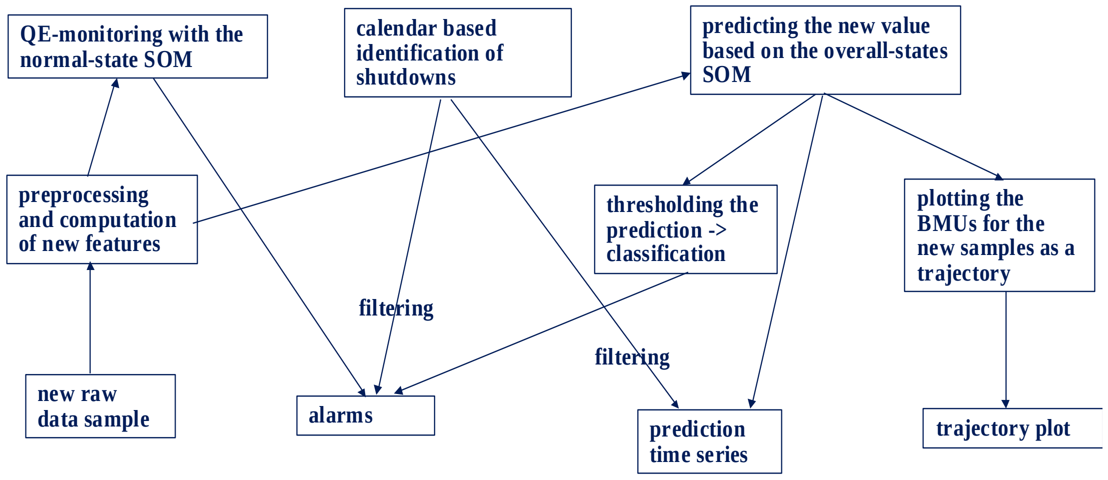
QE-monitoring with the normal-state SOM
calendar based identification of shutdowns
predicting the new value based on the overall-states SOM
preprocessing and computation of new features
thresholding the prediction -> classification
plotting the BMUs for the new samples as a trajectory
filtering
filtering
new raw data sample
alarms
prediction time series
trajectory plot

Figure 6-1 A potential way of combining different models to predict abnormalities

7 Summary

In this report we extend our previous work [Hii00a] aimed at applying neural network methods to detect abnormal conditions in quality measurements of an industrial chemical process. This time we have analyzed a larger set of data and have created various Self Organizing Map (SOM) based models for identifying emerging abnormal states of the process and for predicting the future values of some target variables.

It has been found that only short term predictions are possible with the available process data. Even for short term predictions the accuracy of the predictions is not perfect - the best models reach a level of 80-90% classification accuracy in predicting abnormal situations in advance. So using these methods one can not totally avoid the occasional false alarms and undetected abnormalities.

References

[Alan91]

Alander, J.T., Frisk, M., Holmström, L., Hämäläinen, A., Tuominen, J., Process error detection using self-organizing feature maps, in Proceedings of the Conference on Artificial Neural Networks, pp. 1229-1230, 1991.

[Alho99]

Alhoniemi, E., Hollmen, J., Simula, O., Vesanto, J., Process Monitoring and Modeling using the Self-Organizing Map, Integrated Computer Aided Engineering, John Wiley & Sons, 1/1999, pp. 3-14.

[Hii00a]

Hiirsalmi, Mikko, Prediction of quality indicators based on measurement history, MODUS-Project Waste Water Case Study, Research Report TTE1-2000-28, VTT Information Technology, October 2000, Espoo, Finland, 35 p.

[Hii00b]

Hiirsalmi, Mikko, Method feasibility Study: Bayesian Networks, MODUS-Project Waste Water Case Study, Research Report TTE1-2000-29, VTT Information Technology, October 2000, Espoo, Finland, 34 p.

[Iiva97]

Iivarinen, J., Rauhamaa, J., and Visa, A.. An Adaptive Two-Stage Approach to Classification of Surface Defects. In Proceedings of The 10th Scandinavian Conference on Image Analysis, vol. I, pp. 317-322, Lappeenranta, Finland, June 9-11, 1997.

[Kosk97]

Koskela, T., Varsta, M., Heikkonen, J., and Kaski, K., Time series prediction using RSOM with local linear models. International Journal of Knowledge-Based Intelligent Engineering Systems, 2(1):60-68, 1998. available as Technical Report B15, Helsinki University of Technology, Lab. of Computational Engineering, 1997.

[Moze00] Mozer, M.C., Wolniewicz, R., Grimes, D.B., Johnson, E., Kaushansky, H., Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry, IEEE Transactions on Neural networks, Vol. 11, No. 3, pp. 690-696, May 2000.

[Rint00]

Rinta-Runsala, Esa, Off-Line Analysis and Prototyping of Paper Machine Drive Monitoring System, Modus-Project Case Study ODT2, VTT Information Technology Reasearch Report TTE1-2000-27, pp. 12-16, November, 2000.

[Rita98]

Ritala, R., Paperinvalmistuksen prosessihäiriöiden hallinta, Automaatioväylä, 4, 1998, pp. 11-15.

[Simu99]

Simula, O., Vesanto, J., Alhoniemi, Esa, Hollmen, J., Analysis and Modeling

Of Complex Systems Using the Self-Organizing Map, Chapter in "Neuro-Fuzzy Techniques for Intelligent Information Systems, 1999.

[Taip99]

Taipale, O., Jurva, E., Muuttujien valinta ja mittausaineiston muodostaminen, Tekes Teknologiakatsaus 66/99, 1999.

[Vesa97]

Vesanto, J., Using the SOM and Local Models in Time-Series Prediction, Proceedings of Workshop on Self-Maps (WSOM'97), 1997, pp. 209-214.

[Vesa00]

Vesanto, J., Himberg, J., Alhoniemi, E., Parhankangas, J., SOM Toolbox for Matlab 5, Helsinki University of Technology/CIS Report A57, http://www.cis.hut.fi/projects/somtoolbox, April 2000,

[West00]

West, D., Mangiameli, P., Identifying process conditions in an urban wastewater treatment plant, International Journal of Operations & Production Management, Vol. 20, No. 5, 2000, pp. 573-590.

Appendix A. Statistical descriptors for the preprocessed variables of Ilmas3 line

Statistics

		LAMPO	CODIN	PK3OSUUS	VIRTA3	SAKEUS3	LIETEP3
N	Valid	615	615	615	615	615	615
	Missing	0	0	0	0	0	0
Mean		36.1939	1691.4407	.1829	101.0081	5.0001	6.7899
Std. Deviation		4.8101	356.7145	2.6913	23.4061	.7104	1.1489
Minimum		13.70	25.00	-52.67	-.30	3.31	2.10
Maximum		42.90	2240.00	.81	143.40	7.20	11.11
Percentiles	20	33.3000	1586.6700	.3267	93.3000	4.4307	5.9400
	40	36.2400	1708.5320	.3544	101.6000	4.8147	6.3613
	60	38.2000	1823.4680	.3794	109.6600	5.1473	6.8203
	80	39.9000	1923.2000	.4078	115.4800	5.5800	7.5440

Table A.1 Statistics of the preprocessed variables in Ilmas3 line, A

Statistics

		HAPPI3	HAPPI4	LIEIND3	TYPPI3	LKUORMA3	FKUORMA3
N	Valid	615	615	615	615	615	615
	Missing	0	0	0	0	0	0
Mean		2.6148	4.3358	139.2456	4.3697	.3072	.2208
Std. Deviation		1.7497	1.8721	182.1204	5.4892	9.506E-02	.1713
Minimum		.40	.50	20.71	.50	.00	-.09
Maximum		10.90	10.90	780.23	53.10	.54	1.86
Percentiles	20	1.5000	2.7000	32.0457	2.0000	.2629	.1562
	40	1.9000	3.7000	42.6476	2.3000	.3054	.1885
	60	2.5000	4.6000	71.4000	3.1000	.3396	.2153
	80	3.3800	5.5800	202.0286	4.6000	.3679	.2521

Table A.2 Statistics of the preprocessed variables in Ilmas3 line, B

Statistics

		KIERTOL3	LIKA3	LIUKP3	JSSAK3	CODOUT3	LIUKP3_1
N	Valid	610	615	615	615	615	615
	Missing	5	0	0	0	0	0
Mean		624.9118	19.3294	.6281	68.6634	195.0846	.6273
Std. Deviation		1375.1304	4.7108	.9840	262.6751	46.0297	.9843
Minimum		12.49	4.71	.06	.00	117.00	.06
Maximum		3861.00	29.42	7.70	2866.00	328.00	7.70
Percentiles	20	24.6352	15.4875	.1037	6.8000	154.0000	.1033
	40	31.3300	17.9182	.1900	14.8800	176.0000	.1900
	60	38.8826	20.8232	.4667	26.2500	195.9000	.4640
	80	69.3857	23.6438	.8700	40.7400	235.5336	.8700

Table A.3 Statistics of the preprocessed variables in Ilmas3 line, C

Appendix B. Time series plots of the preprocessed variable values

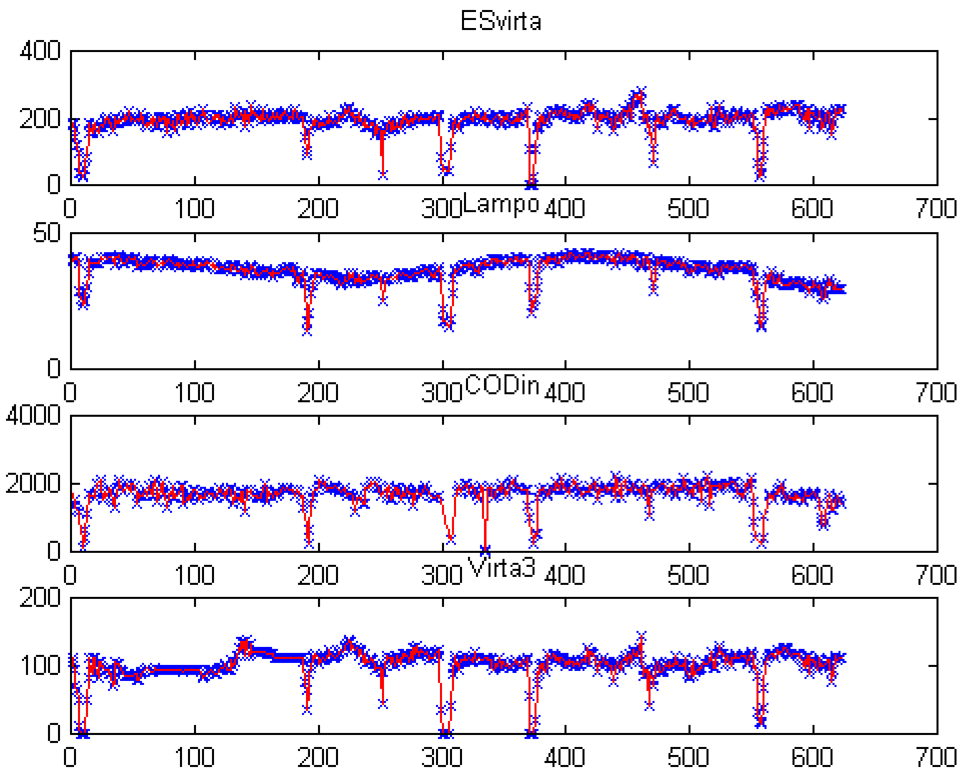

Figure B.1 Time series plots of the raw and preprocessed variables in Ilmas3 line, A. Blue crosses mark the raw measurements and the red line marks the preprocessed value.2

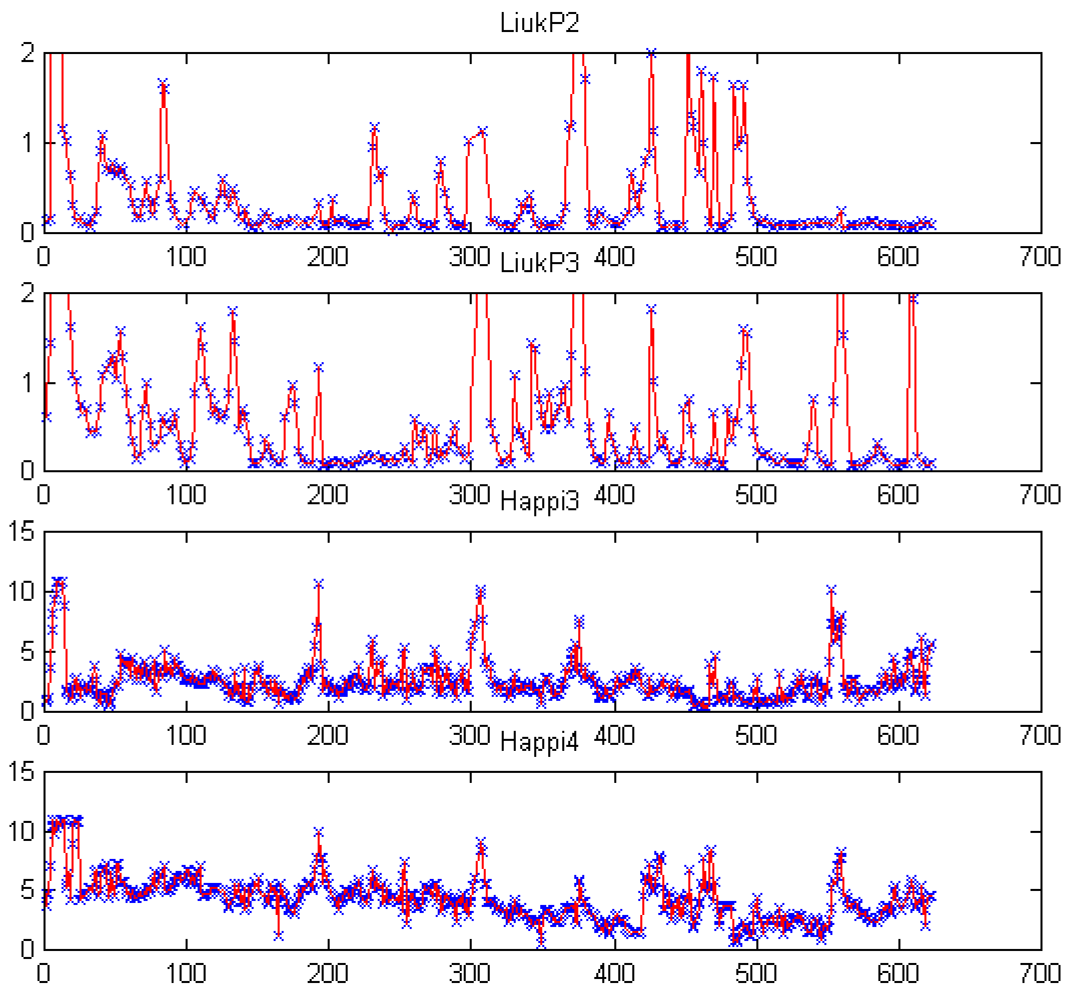

Figure B.2 Time series plots of the raw and preprocessed variables in Ilmas3 line, B. Blue crosses mark the raw measurements and the red line marks the preprocessed value

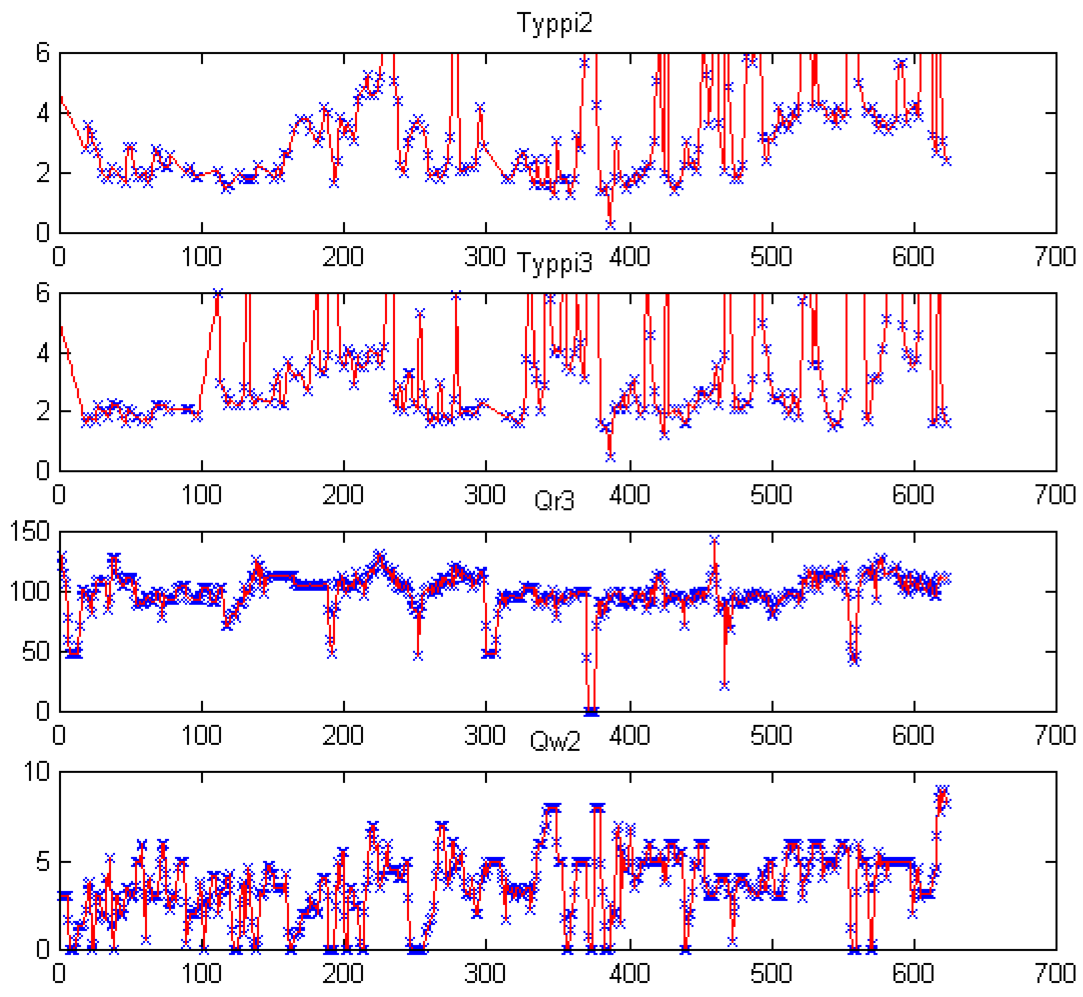

Figure B.3 Time series plots of the raw and preprocessed variables in Ilmas3 line, C. Blue crosses mark the raw measurements and the red line marks the preprocessed value

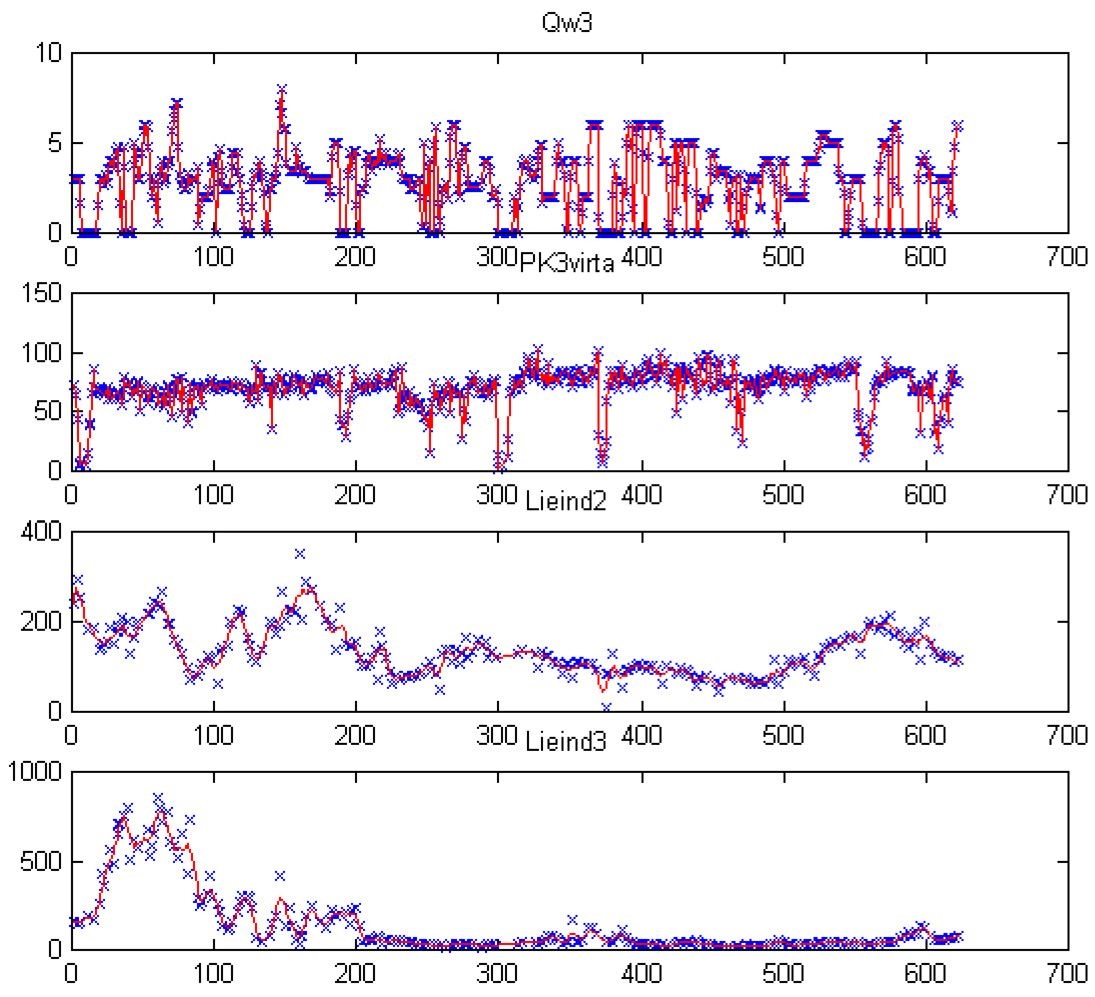

Figure B.4 Time series plots of the raw and preprocessed variables in Ilmas3 line, D. Blue crosses mark the raw measurements and the red line marks the preprocessed value

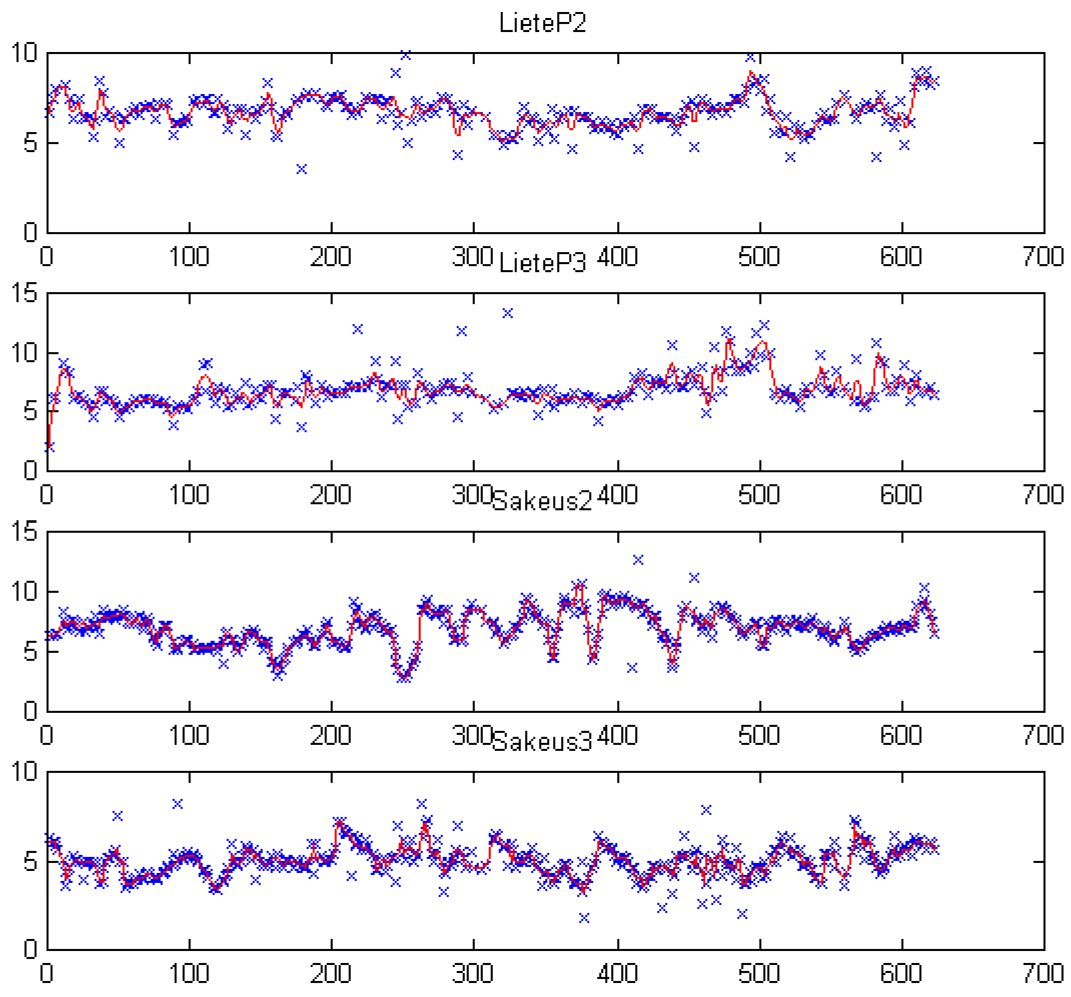

Figure B.5 Time series plots of the raw and preprocessed variables in Ilmas3 line, E. Blue crosses mark the raw measurements and the red line marks the preprocessed value

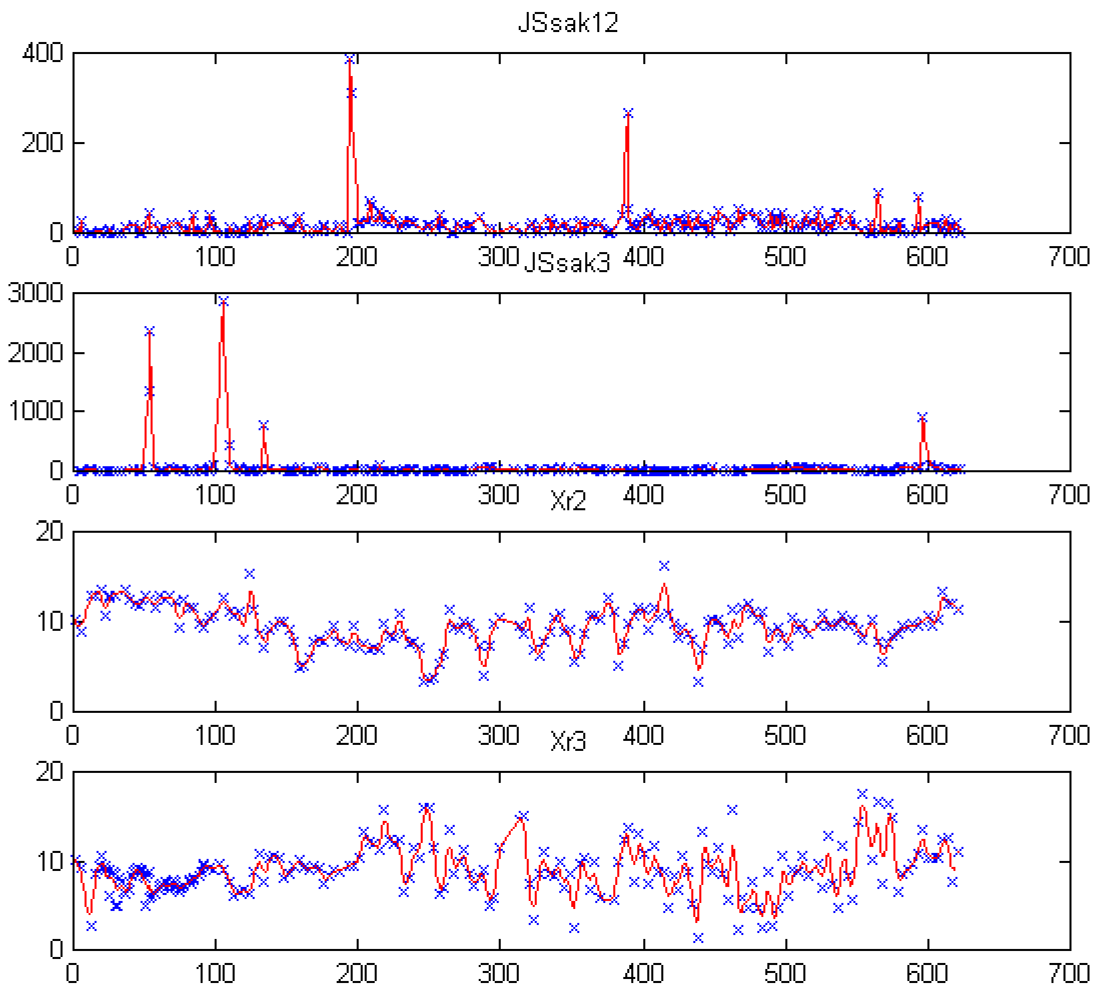

Figure B.6 Time series plots of the raw and preprocessed variables in Ilmas3 line, F. Blue crosses mark the raw measurements and the red line marks the preprocessed value

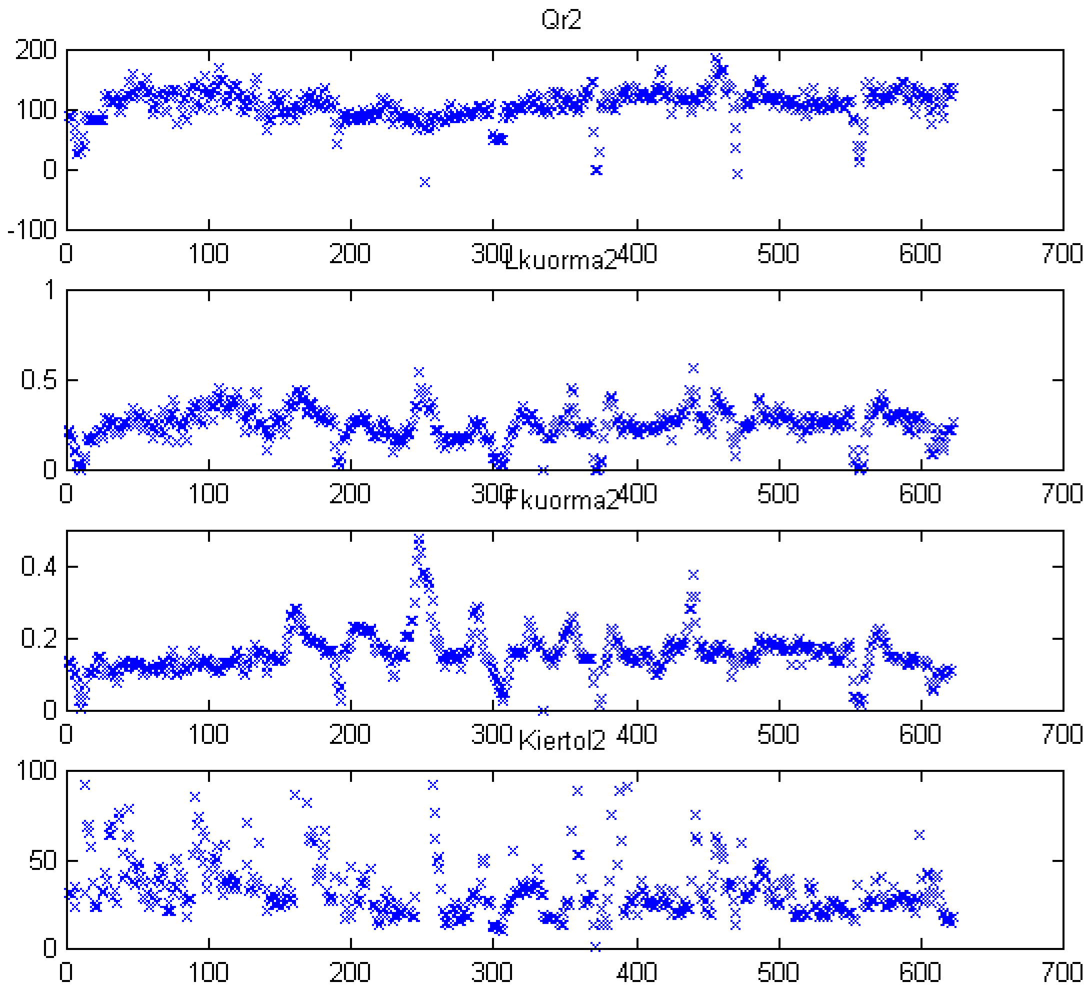

Figure B.7 Time series plots of the raw and preprocessed variables in Ilmas3 line, G. Blue crosses mark the raw measurements and the red line marks the preprocessed value

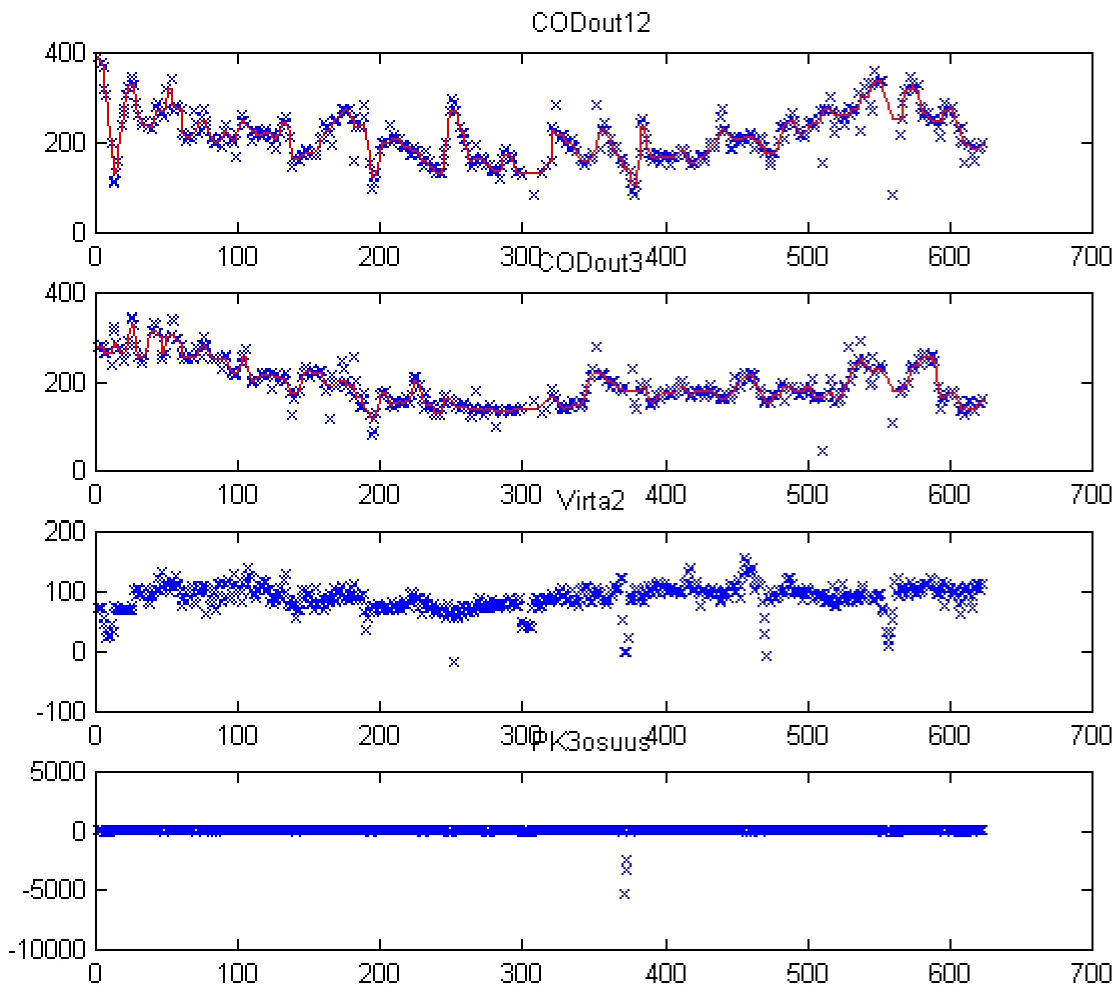

Figure B.8 Time series plots of the raw and preprocessed variables in Ilmas3 line, H. Blue crosses mark the raw measurements and the red line marks the preprocessed value

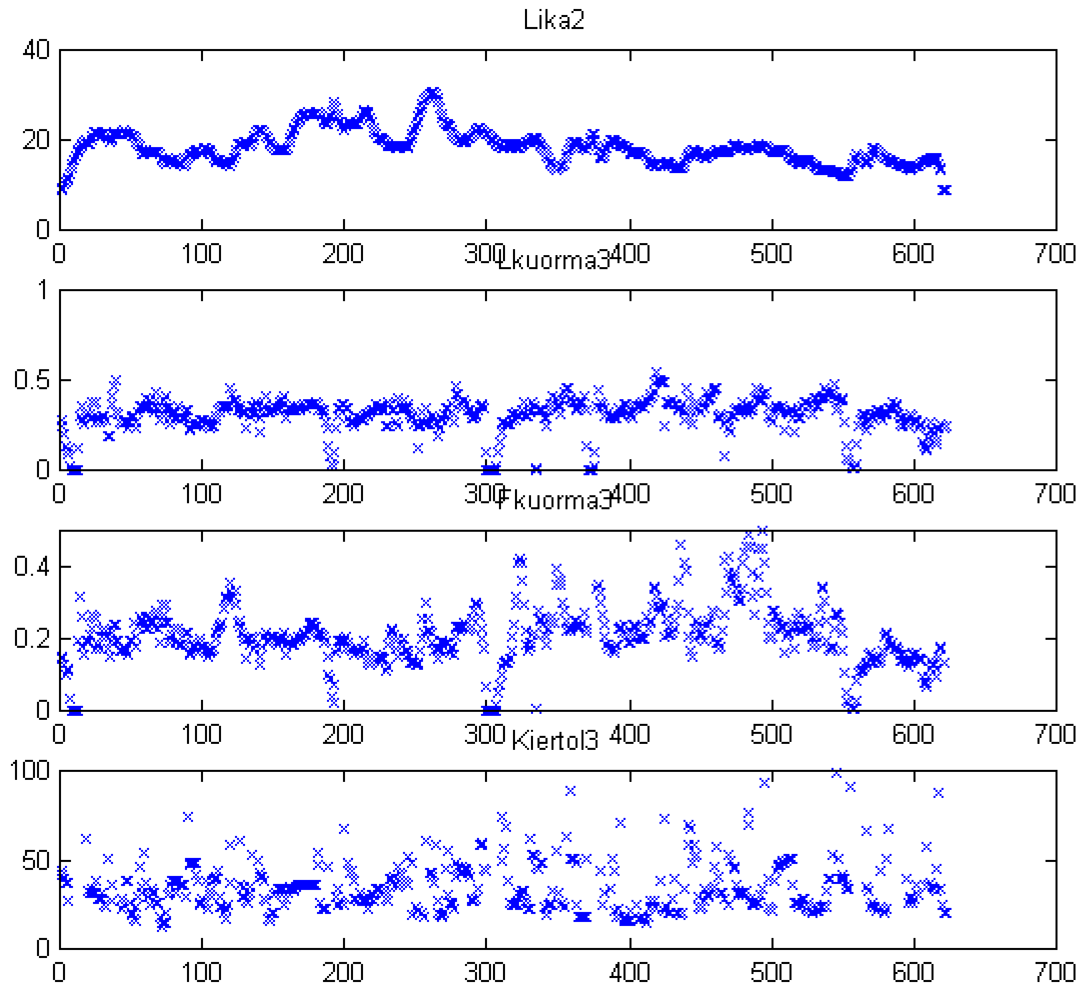

Figure B.9 Time series plots of the raw and preprocessed variables in Ilmas3 line, I. Blue crosses mark the raw measurements and the red line marks the preprocessed value

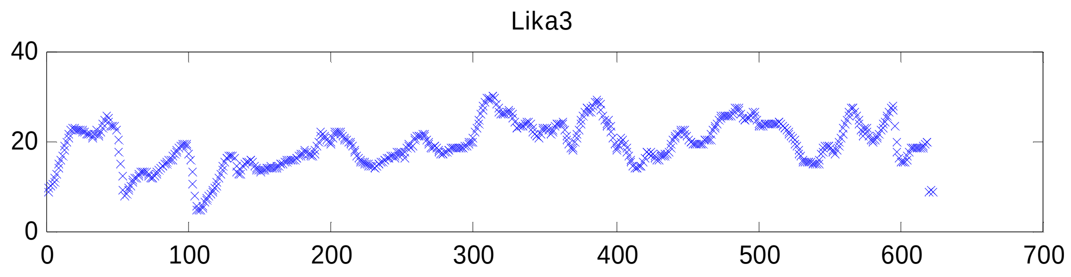

Figure B.10 Time series plots of the raw and preprocessed variables in Ilmas3 line, J. Blue crosses mark the raw measurements and the red line marks the preprocessed value

Appendix C: Correlation analysis results

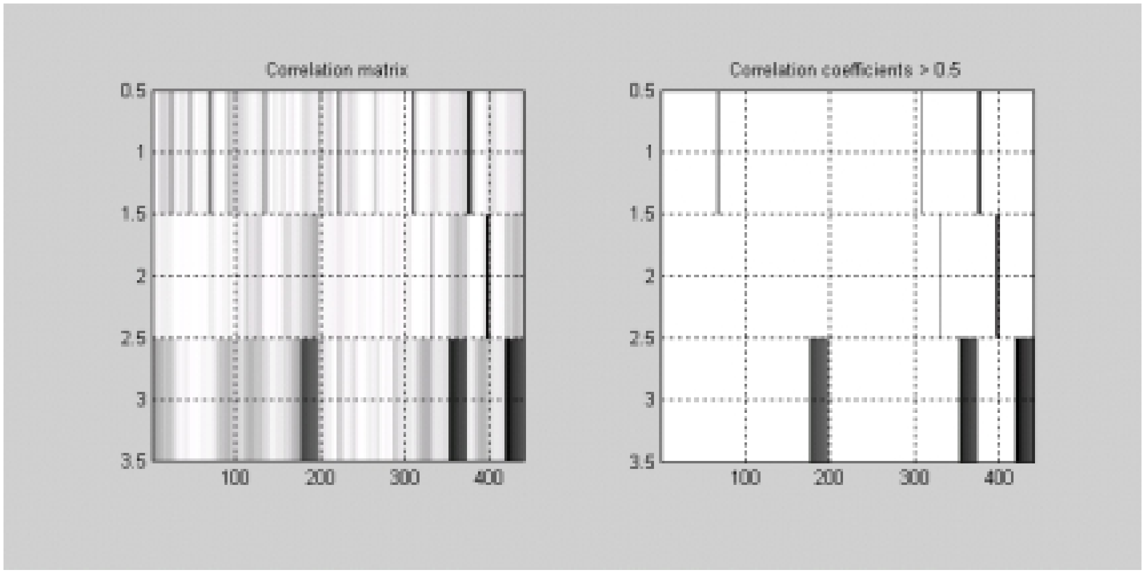

Figure C.1 Correlation matrix for the variables in Ilmas3 line using 21 day delay line reporting only the target variablesto be predicted

Listing C.1 Correlation report for given variables with threshold >= 0.50.

LiukP3_F3 (#18):
- Virta3 (#4), with lags = (0-2).
- LiukP3 (#15), with lags = (0).
- LiukP3_F3 (#18), with lags = (0-3).

JSsak3_F3 (#19):
- JSsak3 (#16), with lags = (0).
- JSsak3_F3 (#19), with lags = (0-3).

CODout3_F3 (#20):
- Lieind3 (#9), with lags = (0-21).
- CODout3 (#17), with lags = (0-21).
- CODout3_F3 (#20), with lags = (0-21).

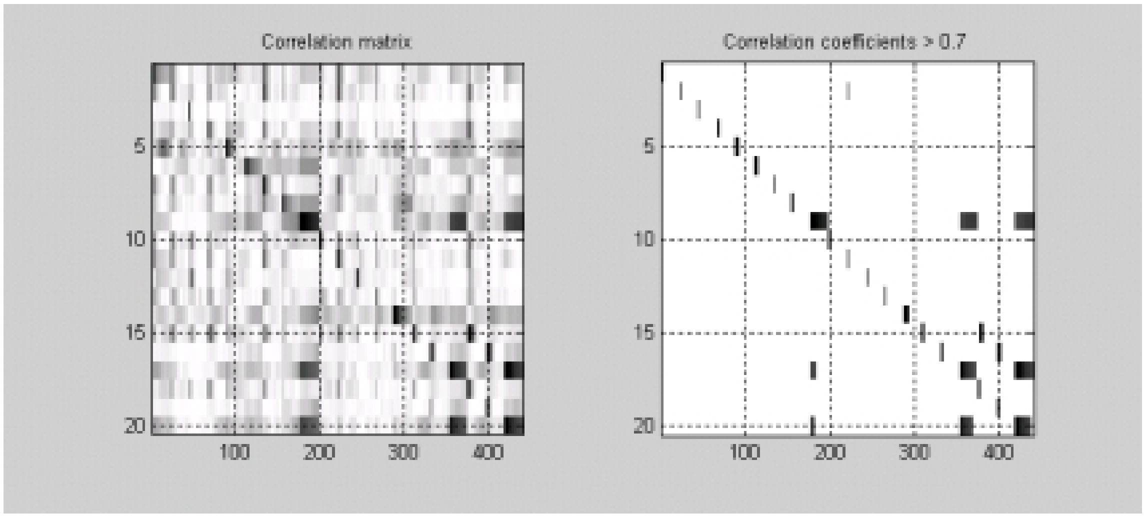

Figure C.2 Correlation matrix for the variables in Ilmas3 line using 21 day delay line

Listing C.2 Correlation report for given variables with threshold >= 0.70.

Lampo (#1):
    Lampo (#1), with lags = (0-3).

CODin (#2):
    CODin (#2), with lags = (0-1).
    Happi3 (#7), with lags = (0).
    Lkuorma3 (#11), with lags = (0).

PK3osuus (#3):
    PK3osuus (#3), with lags = (0).

Virta3 (#4):
    Virta3 (#4), with lags = (0-2).

Sakeus3 (#5):
    Sakeus3 (#5), with lags = (0-3).

LieteP3 (#6):
    LieteP3 (#6), with lags = (0-4).

Happi3 (#7):
    CODin (#2), with lags = (0).
    Happi3 (#7), with lags = (0-1).

Happi4 (#8):
    Happi4 (#8), with lags = (0-2).

Lieind3 (#9):
    Lieind3 (#9), with lags = (0-21).
    CODout3 (#17), with lags = (0-21).

CODout3_F3 (#20), with lags = (0-21).

Typpi3 (#10):
Typpi3 (#10), with lags = (0-2).

Lkuorma3 (#11):
CODin (#2), with lags = (0).
Lkuorma3 (#11), with lags = (0-1).

Fkuorma3 (#12):
Fkuorma3 (#12), with lags = (0-1).

Kiertol3 (#13):
Kiertol3 (#13), with lags = (0).

Lika3 (#14):
Lika3 (#14), with lags = (0-7).

LiukP3 (#15):
LiukP3 (#15), with lags = (0-2).
LiukP3_F3 (#18), with lags = (1-5).

JSsak3 (#16):
JSsak3 (#16), with lags = (0-2).
JSsak3_F3 (#19), with lags = (1-5).

CODout3 (#17):
Lieind3 (#9), with lags = (0-7).
CODout3 (#17), with lags = (0-19).
CODout3_F3 (#20), with lags = (0-21).

LiukP3_F3 (#18):
LiukP3_F3 (#18), with lags = (0-2).

JSsak3_F3 (#19):
JSsak3_F3 (#19), with lags = (0-2).

CODout3_F3 (#20):
Lieind3 (#9), with lags = (0-5).
CODout3 (#17), with lags = (0-15).
CODout3_F3 (#20), with lags = (0-18).

Appendix D: Statistical independency detection results

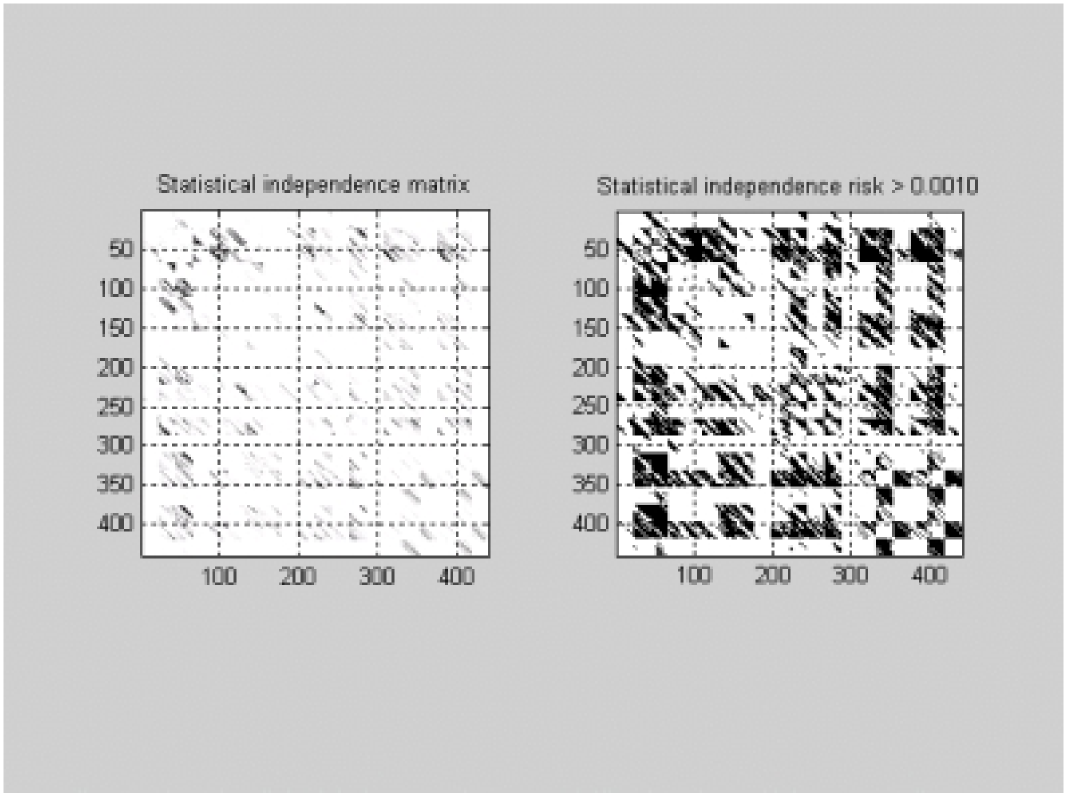

Figure D.1 Statistical independency plotshowing the detected independencies on the given risk level

Listing D.1 Statistical dependency report for given variables with threshold <= 0.0010.

Lampo (#1):
- Lampo (#1), with lags = (0-21).
- CODin (#2), with lags = (0-10).
- PK3osuus (#3), with lags = (0-7, 9-20).
- Virta3 (#4), with lags = (0-21).
- Sakeus3 (#5), with lags = (0-21).
- LieteP3 (#6), with lags = (0-21).
- Happi3 (#7), with lags = (0-21).
- Happi4 (#8), with lags = (0-21).
- Lieind3 (#9), with lags = (0-21).
- Typpi3 (#10), with lags = (0-21).
- Lkuorma3 (#11), with lags = (0-8, 11).
- Fkuorma3 (#12), with lags = (0-21).
- Kiertol3 (#13), with lags = (0-8, 11-18).
- Lika3 (#14), with lags = (0-21).
- LiukP3 (#15), with lags = (0-21).
- JSsak3 (#16), with lags = (0-21).

CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).
JSsak3_F3 (#19), with lags = (0-21).
CODout3_F3 (#20), with lags = (0-21).

CODin (#2):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (0-8, 16-17).
PK3osuus (#3), with lags = (0-11, 14-15).
Virta3 (#4), with lags = (0-4, 13, 16-17).
Sakeus3 (#5), with lags = (17, 21).
LieteP3 (#6), with lags = (5-6, 13, 19-21).
Happi3 (#7), with lags = (0-7, 14, 18-19, 21).
Happi4 (#8), with lags = (0-3, 9-18, 21).
Lieind3 (#9), with lags = (0-6, 16-17, 19-21).
Typpi3 (#10), with lags = (0-2).
Lkuorma3 (#11), with lags = (0-3, 20).
Fkuorma3 (#12), with lags = (0-16).
Kiertol3 (#13), with lags = (0-1).
Lika3 (#14), with lags = (0-11, 15).
LiukP3 (#15), with lags = (0-4, 6, 19).
JSsak3 (#16), with lags = (0-1, 5, 14, 17-21).
CODout3 (#17), with lags = (0-5, 7-21).
LiukP3_F3 (#18), with lags = (0-7, 9).
JSsak3_F3 (#19), with lags = (2-4, 8, 17, 20-21).
CODout3_F3 (#20), with lags = (0-8, 10-21).

PK3osuus (#3):
Lampo (#1), with lags = (0-4, 14-18).
CODin (#2), with lags = (0-9, 11, 18-19).
PK3osuus (#3), with lags = (0-12, 16).
Virta3 (#4), with lags = (0-4, 15).
LieteP3 (#6), with lags = (7-12).
Happi3 (#7), with lags = (0-4, 6-7, 19).
Happi4 (#8), with lags = (0-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0).
Lkuorma3 (#11), with lags = (0-1, 5, 17).
Fkuorma3 (#12), with lags = (0-13).
Kiertol3 (#13), with lags = (0, 11, 14).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (18).
JSsak3 (#16), with lags = (0-2, 16-21).
CODout3 (#17), with lags = (0-7, 9-21).
LiukP3_F3 (#18), with lags = (2, 21).
JSsak3_F3 (#19), with lags = (2-5, 19-21).
CODout3_F3 (#20), with lags = (0-10, 12-21).

Virta3 (#4):
Lampo (#1), with lags = (0-21).

CODin (#2), with lags = (0-4, 6).
PK3osuus (#3), with lags = (0-4, 6).
Virta3 (#4), with lags = (0-17, 19, 21).
Sakeus3 (#5), with lags = (0-21).
LieteP3 (#6), with lags = (0-21).
Happi3 (#7), with lags = (0-5).
Happi4 (#8), with lags = (0-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-4, 6-21).
Lkuorma3 (#11), with lags = (0-5, 10-11, 16-17).
Fkuorma3 (#12), with lags = (0-20).
Kiertol3 (#13), with lags = (5).
Lika3 (#14), with lags = (0-5, 13-21).
LiukP3 (#15), with lags = (0-21).
JSsak3 (#16), with lags = (0-4, 7, 11-15, 17-18).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).
JSsak3_F3 (#19), with lags = (3-7, 10, 14-18, 20-21).
CODout3_F3 (#20), with lags = (0-21).

Sakeus3 (#5):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (11-21).
PK3osuus (#3), with lags = (13-16).
Virta3 (#4), with lags = (0-18).
Sakeus3 (#5), with lags = (0-21).
LieteP3 (#6), with lags = (0-4, 8-21).
Happi3 (#7), with lags = (0-7, 13-21).
Happi4 (#8), with lags = (0-11, 13-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-5, 9-11, 13-21).
Lkuorma3 (#11), with lags = (0-20).
Fkuorma3 (#12), with lags = (0-21).
Kiertol3 (#13), with lags = (0, 2-21).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (0-21).
JSsak3 (#16), with lags = (8-15, 18-20).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).
JSsak3_F3 (#19), with lags = (0-1, 11-18, 21).
CODout3_F3 (#20), with lags = (0-21).

LieteP3 (#6):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (9, 13-21).
PK3osuus (#3), with lags = (11-13, 19-21).
Virta3 (#4), with lags = (0-21).
Sakeus3 (#5), with lags = (0-21).
LieteP3 (#6), with lags = (0-21).
Happi3 (#7), with lags = (0-21).

Happi4 (#8), with lags = (0-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-21).
Lkuorma3 (#11), with lags = (0-1, 4-5, 12, 15, 19-20).
Fkuorma3 (#12), with lags = (0-21).
Kiertol3 (#13), with lags = (0, 8-13).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (0-21).
JSsak3 (#16), with lags = (0, 12-18, 21).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).
JSsak3_F3 (#19), with lags = (0-1, 3, 15-21).
CODout3_F3 (#20), with lags = (0-21).

Happi3 (#7):
Lampo (#1), with lags = (0-5, 8, 16, 18).
CODin (#2), with lags = (0-5, 16).
PK3osuus (#3), with lags = (0-1, 5, 8-9, 15).
Virta3 (#4), with lags = (0-4, 8).
Sakeus3 (#5), with lags = (0-14, 16-17).
LieteP3 (#6), with lags = (0-21).
Happi3 (#7), with lags = (0-21).
Happi4 (#8), with lags = (0-7).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-15, 19-21).
Lkuorma3 (#11), with lags = (0-4, 20-21).
Fkuorma3 (#12), with lags = (0-21).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (0, 14-17, 21).
JSsak3 (#16), with lags = (0, 5, 7, 11-19).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-3, 17-20).
JSsak3_F3 (#19), with lags = (1-3, 8, 10, 14-21).
CODout3_F3 (#20), with lags = (0-21).

Happi4 (#8):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (0-18).
PK3osuus (#3), with lags = (0-21).
Virta3 (#4), with lags = (0-7, 14-21).
Sakeus3 (#5), with lags = (0-3, 7-8, 15-21).
LieteP3 (#6), with lags = (0-21).
Happi3 (#7), with lags = (0-21).
Happi4 (#8), with lags = (0-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-21).
Lkuorma3 (#11), with lags = (0-11, 14, 18-21).
Fkuorma3 (#12), with lags = (0-21).
Kiertol3 (#13), with lags = (0-8, 15-18, 20).
Lika3 (#14), with lags = (0-21).

LiukP3 (#15), with lags = (0, 9, 14-19).
JSsak3 (#16), with lags = (0-2, 14-18).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-3, 12, 17-21).
JSsak3_F3 (#19), with lags = (0-5, 17-21).
CODout3_F3 (#20), with lags = (0-21).

Lieind3 (#9):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (0-21).
PK3osuus (#3), with lags = (0-21).
Virta3 (#4), with lags = (0-21).
Sakeus3 (#5), with lags = (0-21).
LieteP3 (#6), with lags = (0-21).
Happi3 (#7), with lags = (0-15, 17-21).
Happi4 (#8), with lags = (0-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-21).
Lkuorma3 (#11), with lags = (3, 5, 16-20).
Fkuorma3 (#12), with lags = (0-21).
Kiertol3 (#13), with lags = (0-21).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (0-21).
JSsak3 (#16), with lags = (0-14, 16-21).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).
JSsak3_F3 (#19), with lags = (0-17, 19-21).
CODout3_F3 (#20), with lags = (0-21).

Typpi3 (#10):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (0-4, 11-13, 15-17).
PK3osuus (#3), with lags = (0-1, 14-18, 21).
Virta3 (#4), with lags = (0-21).
Sakeus3 (#5), with lags = (0-19, 21).
LieteP3 (#6), with lags = (0-12, 14-19, 21).
Happi3 (#7), with lags = (0-19).
Happi4 (#8), with lags = (0-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-21).
Lkuorma3 (#11), with lags = (0-4, 6-16, 19-21).
Fkuorma3 (#12), with lags = (0-19).
Kiertol3 (#13), with lags = (0-14).
Lika3 (#14), with lags = (1-21).
LiukP3 (#15), with lags = (0-21).
JSsak3 (#16), with lags = (0, 2, 21).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).
JSsak3_F3 (#19), with lags = (0-3, 5).
CODout3_F3 (#20), with lags = (0-21).

Lkuorma3 (#11):
- Lampo (#1), with lags = (0-21).
- CODin (#2), with lags = (0-7, 14-20).
- PK3osuus (#3), with lags = (0-1, 6, 8, 11-12, 15-18).
- Virta3 (#4), with lags = (0-3, 7, 14-15, 18-21).
- Sakeus3 (#5), with lags = (0-10, 16, 18).
- LieteP3 (#6), with lags = (0-4).
- Happi3 (#7), with lags = (0-3, 6, 8, 11, 15-21).
- Happi4 (#8), with lags = (0-11, 16-21).
- Lieind3 (#9), with lags = (4, 7-14).
- Typpi3 (#10), with lags = (0, 2-4, 13).
- Lkuorma3 (#11), with lags = (0-10).
- Fkuorma3 (#12), with lags = (0-12, 15-16, 18).
- Kiertol3 (#13), with lags = (0-16).
- Lika3 (#14), with lags = (0-13, 15-21).
- LiukP3 (#15), with lags = (0-1, 14-18).
- JSsak3 (#16), with lags = (0).
- CODout3 (#17), with lags = (0-5, 11-21).
- LiukP3_F3 (#18), with lags = (0-4, 17-21).
- JSsak3_F3 (#19), with lags = (2-3).
- CODout3_F3 (#20), with lags = (0-8, 14-21).

Fkuorma3 (#12):
- Lampo (#1), with lags = (0-21).
- CODin (#2), with lags = (0-18).
- PK3osuus (#3), with lags = (0-4, 6-9, 17-19).
- Virta3 (#4), with lags = (0-21).
- Sakeus3 (#5), with lags = (0-10, 13-17).
- LieteP3 (#6), with lags = (0-21).
- Happi3 (#7), with lags = (0-7, 11-16, 18-21).
- Happi4 (#8), with lags = (0-6, 9-21).
- Lieind3 (#9), with lags = (0-21).
- Typpi3 (#10), with lags = (0-6, 8-12, 18-20).
- Lkuorma3 (#11), with lags = (0-7, 14).
- Fkuorma3 (#12), with lags = (0-21).
- Kiertol3 (#13), with lags = (0-9, 12-14).
- Lika3 (#14), with lags = (0, 2-21).
- LiukP3 (#15), with lags = (0-11, 13-16, 18-21).
- JSsak3 (#16), with lags = (0, 13-17).
- CODout3 (#17), with lags = (0-21).
- LiukP3_F3 (#18), with lags = (0-14, 16-19, 21).
- JSsak3_F3 (#19), with lags = (0-3, 16-20).
- CODout3_F3 (#20), with lags = (0-21).

Kiertol3 (#13):
- Lampo (#1), with lags = (0-11, 13).
- CODin (#2), with lags = (0-8).
- PK3osuus (#3), with lags = (0, 2, 10).
- Virta3 (#4), with lags = (14-21).

Sakeus3 (#5), with lags = (0-8).
LieteP3 (#6), with lags = (0-3, 5-6, 16-19, 21).
Happi3 (#7), with lags = (2-6).
Happi4 (#8), with lags = (0-1, 4-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-3, 16-17).
Lkuorma3 (#11), with lags = (0-3, 16-17, 19).
Fkuorma3 (#12), with lags = (0-4, 7).
Kiertol3 (#13), with lags = (0-5, 9-10).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (0-2).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-5).
CODout3_F3 (#20), with lags = (0-21).

Lika3 (#14):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (0-10, 19).
PK3osuus (#3), with lags = (0-21).
Virta3 (#4), with lags = (0-21).
Sakeus3 (#5), with lags = (0-21).
LieteP3 (#6), with lags = (0-21).
Happi3 (#7), with lags = (0-21).
Happi4 (#8), with lags = (0-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (1-13, 18-21).
Lkuorma3 (#11), with lags = (0-14, 16-21).
Fkuorma3 (#12), with lags = (0-21).
Kiertol3 (#13), with lags = (0-17).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (0-21).
JSsak3 (#16), with lags = (0-21).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).
JSsak3_F3 (#19), with lags = (0-21).
CODout3_F3 (#20), with lags = (0-21).

LiukP3 (#15):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (0-3).
PK3osuus (#3), with lags = (1, 16).
Virta3 (#4), with lags = (0-21).
Sakeus3 (#5), with lags = (0-21).
LieteP3 (#6), with lags = (0-16).
Happi3 (#7), with lags = (0-3, 10-11, 19).
Happi4 (#8), with lags = (0-5, 17-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-8, 13-21).
Lkuorma3 (#11), with lags = (0-7).
Fkuorma3 (#12), with lags = (0-19).

Kiertol3 (#13), with lags = (0-9, 11-12).
Lika3 (#14), with lags = (0-12, 14-21).
LiukP3 (#15), with lags = (0-21).
JSsak3 (#16), with lags = (0-2, 4-6, 10-19, 21).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).
JSsak3_F3 (#19), with lags = (0-5, 7-9, 13-21).
CODout3_F3 (#20), with lags = (0-21).

JSsak3 (#16):
Lampo (#1), with lags = (0-6, 8-21).
CODin (#2), with lags = (0-1, 10-11, 15-19).
PK3osuus (#3), with lags = (0-1).
Virta3 (#4), with lags = (0, 9-16).
Sakeus3 (#5), with lags = (2-6, 8-11).
LieteP3 (#6), with lags = (0, 2-21).
Happi3 (#7), with lags = (0-2, 15).
Happi4 (#8), with lags = (0-5, 14).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-3, 6, 10-11, 17-21).
Lkuorma3 (#11), with lags = (0-1, 11-13).
Fkuorma3 (#12), with lags = (0-3, 6, 10-13, 19-21).
Kiertol3 (#13), with lags = (4, 12-19, 21).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (0-2, 4, 14-16).
JSsak3 (#16), with lags = (0-21).
CODout3 (#17), with lags = (6-7, 11-12, 15-21).
LiukP3_F3 (#18), with lags = (0-5, 7, 17-19).
JSsak3_F3 (#19), with lags = (0-21).
CODout3_F3 (#20), with lags = (9-10, 14-15, 18-21).

CODout3 (#17):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (0-21).
PK3osuus (#3), with lags = (0-21).
Virta3 (#4), with lags = (0-21).
Sakeus3 (#5), with lags = (0-21).
LieteP3 (#6), with lags = (0-21).
Happi3 (#7), with lags = (0-21).
Happi4 (#8), with lags = (0-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-21).
Lkuorma3 (#11), with lags = (0-10, 12-21).
Fkuorma3 (#12), with lags = (0-21).
Kiertol3 (#13), with lags = (0-21).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (0-21).
JSsak3 (#16), with lags = (17, 20).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).

JSsak3_F3 (#19), with lags = (20).
CODout3_F3 (#20), with lags = (0-21).

LiukP3_F3 (#18):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (0-1).
PK3osuus (#3), with lags = (13).
Virta3 (#4), with lags = (0-21).
Sakeus3 (#5), with lags = (0-21).
LieteP3 (#6), with lags = (0-12, 21).
Happi3 (#7), with lags = (0, 7-9, 12, 14, 16-17).
Happi4 (#8), with lags = (0-1, 14-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-5, 10-21).
Lkuorma3 (#11), with lags = (0-4).
Fkuorma3 (#12), with lags = (0-19).
Kiertol3 (#13), with lags = (0-6, 8-9).
Lika3 (#14), with lags = (0-10, 12-21).
LiukP3 (#15), with lags = (0-21).
JSsak3 (#16), with lags = (0-1, 8-16).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).
JSsak3_F3 (#19), with lags = (0-4, 11-19).
CODout3_F3 (#20), with lags = (0-21).

JSsak3_F3 (#19):
Lampo (#1), with lags = (0-3, 5-21).
CODin (#2), with lags = (6-8, 13-16).
Virta3 (#4), with lags = (5-13).
Sakeus3 (#5), with lags = (0-3, 5-8).
LieteP3 (#6), with lags = (0-21).
Happi4 (#8), with lags = (0-2, 11).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-1, 3, 7-9, 14-17, 21).
Lkuorma3 (#11), with lags = (8-10, 20-21).
Fkuorma3 (#12), with lags = (0, 7-10, 16-19, 21).
Kiertol3 (#13), with lags = (1, 3, 7, 9-16, 18).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (0-1, 11-13, 20-21).
JSsak3 (#16), with lags = (0-16, 18-19).
CODout3 (#17), with lags = (3, 6, 12-21).
LiukP3_F3 (#18), with lags = (0-4, 14-16).
JSsak3_F3 (#19), with lags = (0-19, 21).
CODout3_F3 (#20), with lags = (6, 9, 15-21).

CODout3_F3 (#20):
Lampo (#1), with lags = (0-21).
CODin (#2), with lags = (0-21).
PK3osuus (#3), with lags = (0-21).
Virta3 (#4), with lags = (0-21).

Sakeus3 (#5), with lags = (0-21).
LieteP3 (#6), with lags = (0-21).
Happi3 (#7), with lags = (0-21).
Happi4 (#8), with lags = (0-21).
Lieind3 (#9), with lags = (0-21).
Typpi3 (#10), with lags = (0-21).
Lkuorma3 (#11), with lags = (0-7, 11-21).
Fkuorma3 (#12), with lags = (0-21).
Kiertol3 (#13), with lags = (0-6, 9-18).
Lika3 (#14), with lags = (0-21).
LiukP3 (#15), with lags = (0-21).
JSsak3 (#16), with lags = (14-15, 17, 19-20).
CODout3 (#17), with lags = (0-21).
LiukP3_F3 (#18), with lags = (0-21).
JSsak3_F3 (#19), with lags = (17-18, 20).
CODout3_F3 (#20), with lags = (0-21).