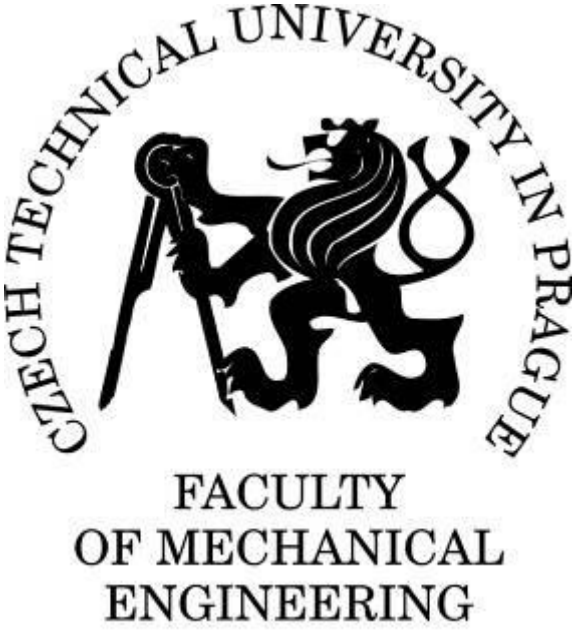
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF MECHANICAL ENGINEERING

Department of Instrumentation and Control Engineering
Division of Automatic Control and Engineering Informatics



Bachelor's Thesis

Data Pre-Processing for Adaptive Modelling of Heating System

Author: Ainur Duisenbayeva
Supervisor: doc. Ing. Ivo Bukovský, Ph.D.

Academic Year: 2012/2013

Statement

I declare that I have worked out this thesis independently assuming that the results of the thesis can be also used at the discretion of the supervisor of the thesis as its co-author. I also agree with the potential publication of the results of the thesis or of its substantial part, provided I will be listed as the co-author.

Prague, ……

Signature……..

Acknowledgement

The author and the supervisor would like to thank to

Energocentrum Plus, s.r.o.

for providing us with measured real data of building heating system that could be used for analysis in this thesis.

The author would also like to thank to her supervisor, doc. Ing. Ivo Bukovský, Ph.D. for his constant guidance, help and encouragement during her work on this thesis.

Abstract

This bachelor's thesis is a case study of artificial data and real data of SISO heating system using correlation analysis plus uncertainty evaluation of input-output data for possible neural network models. Also, possible utilization of coarse-graining technique is studied for data pre-processing. Uncertainty in data is estimated in this work by means of Multiscale False Neighborhood Analysis. Concludingly, appropriateness of configuration of state vector for a prediction interval as for neural network models is validated by both cross-correlation function of heating data and estimation of uncertainty in data.

Contents

Notation

u control input, input temperature
y controlled variable, output temperature
k discrete index of time with constant sampling
r correlation coefficient
n prediction interval

1 Introduction

Many researches are focused on implementation of Computational Intelligence (CI) methods for building heating systems modeling and control. These CI methods are mainly data-based modeling techniques, so they require some amount of measured data. Real-world data is not always suitable for modeling, therefore data pre-processing is step important for data-based modeling methods. Due to pace of development of CI for heating systems and pre-processing techniques for them deep and detailed review is probably impossible, and of course beyond the scope of this thesis. Therefore, section 2 presents brief reviews of both CI methods used in building heating systems and data pre-processing techniques.

One of the objectives behind this thesis is to analyze data by correlation analysis. Pre-processing with correlation analysis is used not only to analyze relationship between data, but also to find time lags of the heating system.

Second objective of mine is to perform correlation analysis with coarse-graining technique. Coarse-graining is presented in such a way, that it is used as filtering and downsampling method.

The third goal of this work is to perform Multiscale False Neighborhood analysis in order to find uncertainty in data. The analysis helps to evaluate time lag of the heating system at which number of uncertainty in data is minimal, and in alliance with correlation analysis it fulfills the last objective of the thesis, which is to find optimal prediction interval for configuration of the state vector for possible neural network model of heating data.

2 State of the Art

2.1 Computational Intelligence Methods for Heating Systems Modeling and Control

Modeling and control of energy efficient heating system has been the main focus of many researches over the past years. The main topics of these researches are: application of computational intelligence (Neural Network, Fuzzy Logic), predictive models and non-linear heating systems. Despite of all developments PID controller is still the most popular control method of heating systems. The other popular method of control is Model Predictive Control (MPC) [4]. Prediction of output temperature and weather forecast helps to set optimal input temperature, which can significantly decrease energy consumption and therefore MPC said to be very effective.

Computational Intelligence (CI) is a system of control capable of "understanding" and training with regard to control objects, disturbances of the environment and working conditions. The main difference of the Intelligent Systems is its mechanism of knowledge processing. CI application in heating systems can be divided in two types: CI used as predictive model and CI used for control of heating system. In this subsection literature review on CI application in heating systems and in Heating Ventilation and Air Conditioning (HVAC) systems is presented.

For accurate thermal dynamic analysis of heating system large number of equations with complex mathematical operations may be needed, still this analysis would not be accurate. In other words, it is almost impossible due to complexity and time consumption. However, there are two computational intelligence techniques that can help to avoid those obstacles and analyze complex response of heating system. Those techniques are Fuzzy Logic and Artificial Neural Network.

Fuzzy Logic (FL) theory is based on concept of fuzzy sets that represent degree of membership and can be equal to number between [0, 1]. FL is used in control of plants that are hard to mathematically describe and where human operator can provide qualitative control for the system, therefore is it popular in heating control. There are two types of FL inference systems Mamdami and Sugeno types. Fuzzy Logic has been used in building heating system control for many years. The objective of fuzzy logic model in heating system is to estimate the heat requirement of the house. Simple if-then rules work for estimation of required heat.

Artificial Neural Network (ANN) is mathematical tool, which is used for mapping of input vector into output vector. ANN consists of neurons in multiple layers. There are various types of ANN and several training techniques of them.

The most common use of CI in heating systems is as a predictive model. Artificial Neural Network in control of a building can be used as a tool to predict indoor temperature and energy consumption of the building. ANN is useful as prediction of off and on time for HVAC system [12]. Prediction of temperature in office rooms based on weather forecast helps to find optimal start time of HVAC for thermal comfort of occupants in working hours and stop time for non-working hours for energy saving. Work [9] presents Adaptive ANN as a prediction model. It has a big advantage in comparison with static ANN. While static model is fixed and requires large amount of data in advance, dynamic ANN can adapt itself when new information is available. Training of dynamic Neural Network is done in two ways: the accumulative training, and the sliding window training. Researchers use ANN model with one hidden layer. For large amount of data work [9] recommends to use Gradient Descent algorithm of training, while for smaller number of data Levenberg-Marquart (LM) algorithm is more effective.

Computational Intelligence methods for HVAC control is recent years proved to be more energy efficient and provide more thermal comfort in comparison with traditional control methods (On/Off control and PID). Intelligent control techniques can be used as a technique for improving traditional controllers work and as a separate intelligent controller. Main goal of most of CI implementation in HVAC is reduction of energy consumption, while providing occupants thermal comfort. Work [10] makes prediction and control of important for human thermal comfort conditions indoor temperature, humidity and PMV (predicted mean vote) using ANN. Research uses ANN with one hidden layer and seventeen neurons, and LM training algorithm. ANN based predictive control it is more efficient in thermal control of residential building, than conventional controllers. Another advantage of the predictive nature of ANN is reduced magnitude of overshoots and undershoots.

Work [7] uses FL model for evaluation of energy requirements and as fuzzy controller.

More advanced use of intelligent method in HVAC control is Neuro-Fuzzy model control. Neuro-Fuzzy method of control is based on human like learning ability of NN and human like thinking way of Fuzzy Logic. Research [11] focused on developing emotional learning algorithm based on Neuro-Fuzzy learning methods for control of Cooling-Heating systems. Typically, neuro-fuzzy model can be divided in two types: Takagi Sugeno's fuzzy inference system and Neuro-Fuzzy local linear model. In paper [11] first type, Takagi Sugeno model has been used. This type of Neuro-Fuzzy model produces two emotional signals (error and derivation of error), deffuzification of this signal is made by mean of center method. Even though, intelligence method is prevailing topic for optimal control and modeling of heating system, it still has to be studied deeper in order to be customized in real life.

2.2 Data Pre-processing Techniques for Heating Processes

In this subsection we outline most important steps in data pre-processing for CI modeling of heating systems.

As mentioned in subsection above, computational intelligence (CI), such as neural network, is widely applied in controlling and modeling of heating systems. Since CI is data-based modeling technique data pre-processing has huge impact on quality of obtained result. In fact, properly prepared data makes computation of CI easier and solutions more accurate. There is no clear answer on what steps data pre-processing should include; it can include cleaning, normalization, transformation, selection and etc. However, there are three main steps that always present in data pre-processing:

- Data cleaning
- Data transformation (sampling)
- Data selection

The aim of data pre-processing is to provide quality data for future processing applications. Data pre-processing for any CI model analysis is complicated task, but Neural Networks have some special requirements. An integrated way of data pre-processing used in work [14] has been made in two steps: data inspection, data processing. First step identifies impurities and errors in data that may affect the model. Second step deals with them, by use of several techniques, such as sampling, filtering, normalization, repairing. Those steps can be used not only for Neural Networks, but for any kind of CI analysis. Data selection in research is handled by dividing data to forms of the clusters of sample observations. Data transformation proposed to be done by correlation of the data in case of dealing with missing values. The idea is based on assumption that there are relations between data tuples over all range of data.

Data cleaning step in data pre-processing is noise reduction step. Usually for data denoising low-pas filters are used, such as moving average filter. For both online and offline data denoising simple linear filters used, as well as linear regression technique [14]. These filters are easier to implement, than any other [16] low-pass filters, but with some loses in accuracy of filtering. The other filtering used in work is

Correlation analysis can be applied for both heating systems prediction model construction and data pre-processing for CI modeling.

Correlation analysis has been used for constructing prediction model for building heating systems for many years. Prediction is based on correlation between energy usage and temperature (indoor or weather temperature). Work [15] uses correlation analysis to handle heating and cooling calculations by dealing with internal and solar gains. This method of prediction was popular before CI implementation in heating system prediction models.

Paper [17] stated importance of data preprocessing for fuzzy logic prediction model. Preprocessing based on correlation analysis is introduced as new and effective pre-processing technique for predictor modeling. According to autocorrelation analysis candidates of difference intervals for construction of predictor are chosen. This step helps to minimize prediction error of fuzzy logic predictor. Results of prediction showed that prediction made with preprocessed data is better than with original data. Also cross-correlation function is proposed to be used as compensation error detection method. The error corresponding to maximum cross-correlation coefficient is said to be compensation error.

There are many more data pre-processing techniques, and which of them to use depends on data and CI technique used for modeling.

3 Used Approaches

After data from building heating systems has been measured it must be examined to find any unusual characteristics, complex relationship between variables, therefore data pre-processing is needed. In our case data pre-processing is done in three steps. The first step is inspection of the data, where correlation between data is determined. Second step includes coarse-graining technique (filtering, checking sample rate and sample length of data, and resampling of data, if needed). Third and final step is uncertainty evaluation and finding optimal prediction interval for chosen state vector for possible Neural Network modeling.

3.1 Correlation Analysis

Correlation analysis provides information about the strength of a linear relationship between two vectors of data. The analysis defines if a set of input variables is able to estimate a set of output variables. Correlation coefficient defines the degree of relationship between variables.

Pearson's correlation coefficient was introduced by Karl Pearson in 1895 as sum of cross-product of two variables. The formula (2.1) he developed is still widely used in correlation analysis. The absolute value of denominator is always bigger or equal to absolute value of the numerator; therefore correlation coefficient changes from -1 (strong negative linear relationship) to 1(strong positive linear relationship). If it equals 0 there is no linear dependence between variables.

$$r = \frac{\sum_{i=1}^{n}(u_i - \bar{u})(y_i - \bar{y})}{[\sum_{i=1}^{n}(u_i - \bar{u})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2]^{\frac{1}{2}}} \qquad (1)$$

where $\bar{u} = \frac{1}{n}\sum_{i=1}^{n} u_i$ and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

Usually two time series are shifted in time relative to one another. These shifts are time lags that are very important characteristic of heating system. There are several types of time lags in HVAC system, but for heating system distance-velocity time lag can be considered as most significant of them. The distance-velocity lag is the time between a temperature being set on heater and sent to system and the system starting to respond. Thus, just using correlation coefficient will not give us a proper result; instead we use lagged correlation or cross-correlation function:

$$\mathrm{r}(lag) = \frac{\sum_{i=1}^{n}(u_i - \bar{u})(y_{i+lag} - \bar{y})}{[\sum_{i=1}^{n}(u_i - \bar{u})^2 \sum_{i=1}^{n}(y_{i+lag} - \bar{y})^2]^{\frac{1}{2}}} \qquad (2)$$

where and r(lag) is cross-correlation function between variables at some time lag. Cross-correlation function is correlation between time series shifted against one another by some time lag. [7]

Cross-correlation function is not symmetrical function, thus value of cross-correlation at time t=lag is not equal to value of cross-correlation function at time t=-lag. In cross-correlation function one of the variables can be considered as lead variable and other is as lagged. If we assume that u is lead variable, than y is shifted variable, which depends on u, and value of maximum correlation coefficient will be at positive time lag. It is well known that correlation

between time series does not prove causality of the system, however if system is causal then correlation between time series most certainly will be strong and positive. [17]

3.2 Coarse Graining Technique (CGT)

In Reference [5] new time series analysis for biological data had been introduced Multiscale Entropy Analysis (MSEA). It is a technique for complexity measurement based on Entropy calculation at various time scales.

Since sample entropy and approximate entropy analysis are performed on single scale, researches applied Coarse Grainning Technique to perform entropy analysis on multiple scales and compare the results. For a given time series $\{x_1,...,x_i,...,x_N\}$ we construct coarse-graining time series $\{y^{(\tau)}\}$, where τ is scale factor. According to equation (3) coarse-graining of time series is made in two steps: a) original time series is divided into windows with length τ; b) the data points are averaged inside each window. These two steps can be considered as resampling and filtering.

$$y_j^{\tau} = \frac{1}{\tau} \sum_{i=(j-1)\tau+1}^{j\tau} x_i, \quad 1 \le j \le \frac{N}{\tau} \qquad (3)$$

The length of each coarse-grained series is length of the original time series divided by scale factor. For time series with scale factor 1, the coarse-grained time-series is the original time series. Coarse-graining technique for data pre-processing can be applied as filtering and sampling method, since in case of Multiscale Entropy Analysis data after applying CGT became smother and reduced in scale.

3.2.1 Correlation analysis and CGT

As can be seen from previous subsection Coarse Graining has been successfully used as smoothing and rescaling technique in MSE analysis. Depending on applied time scale the length of the time series changed. Inspired by Coarse Graining Technique I use similar approach for noise reduction and rescaling. Coarse Graining Technique is used in this work in such way, that I use principle of averaging in coarse-graining as filtering of the signal and instead of dividing time series by time scale factor τ, I use downsampling of the data, as a way to reduce amount of data.

As a filtering I use Moving Average filter, which is average of every point in data equation (4).

$$u[i] = \frac{1}{N} \sum_{j=0}^{M-1} x[i+j] \qquad (4)$$

where u[] is filtered data, x[] is input data, N is number of data used in moving average.

Most data is affected to some extend by noise that can be removed from data by some filter. In this work Moving Average filter is used. It has been shown in reference [6] that Moving Average is optimal filtering technique for reducing noise and keeping the sharpest edges, despite the fact that it is the easiest one.

Coarse Graining formula (3) uses scale factor τ to change time series scale, I use downsampling to change sampling rate of the data. Reduction of the data rate and data size by some integer M is called downsampling. M is called downsampling rate and it is an integer number, greater than unity. [7]

Downsampling used in this work is made by picking out every M-th sample

$$h(k) = g(Mk) \qquad (5)$$

3.3 Multiscale False Neighborhood Analysis

When modeling CI on raw data, i.e. unprepared experimentally measured data, expected result from predictive model often cannot be achieved. This is because data may contain incorrect and even inconsistent points for one reason or another. These uncertainties for example in neural network can affect on training speed, on the magnitude of training error and even can lead to inefficiency of built CI model. These points should be eliminated from the data, or the sampling and time lags for training should be picked out in such a way, that these points would be minimal. Multiscale False Neighborhood Analysis is used in this work for dealing with uncertainties in heating data.

3.3.1 False Neighborhood Analysis

False Neighborhood Analysis (FNA) was introduced by Kernel for estimation of embedding distance in Reference [13]. The basic goal of research is to determine appropriate embedding dimension for state space reconstruction for nonlinear time series analysis, this is done by FNN method. The principle is to define Euclidean distance between some point $\vec{y}_i$ and its nearest neighbor $\overrightarrow{y_j}$, as dimension of space increases distance between points should not change if they are nearest neighbors, otherwise these points are called false neighbors.

Necessary condition of uniqueness of mapping is the core idea of false neighborhood analysis

$$f(x) = y \qquad (6) \ ,$$

where x is input vector of state of the system, y is output of the system, and f is mapping function.

$$\text{If} \quad \|x_1 - x_2\| > d_x \text{ and } \|y_1 - y_2\| > d_y$$
$$\text{then } x_1 \text{ and } x_2 \text{ are False Neighbors} \qquad (7)$$

where x_1 and x_2 are states of the system, y_1 and y_2 are system outputs, d_x and d_y are absolute radii (boundaries) describing similar or distinct system states. Equation (7) describes fundamental rule of False Neighbor Analysis. [1][2]

3.3.2 Multiscale False Neighbor Matrix

Some advanced computational techniques, such as neural network, requires proper defined state vector as an input parameter. For estimation of such state vector Multiscale False Neighbors

analysis was proposed in work [1][2][3]. MSFNA defined in research as a tool for comparing of different proposed state vectors x by defining level of uncertainty in data.

Evaluation of uncertainty in data can be done by False Neighbors technique. In MSFNA, instead of calculating false neighbors for single radii, as it is described in work [14], false neighbors in MSFNA are calculated for a whole range of radii. General state vector model is can be represented as follows:

$$x(k) = \begin{bmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k-n_y) \\ u(k) \\ u(k-1) \\ \vdots \\ u(k-n_u) \end{bmatrix} \qquad (8)$$

As can be seen from equation (2.6) state vector requires some time lags n_u and n_y. Defining proper time lags would help to minimize number of false neighbors.

$$f(x(k)) = y(k+n) \qquad (9)$$

In (9), modeled variable y need to be predicted from estimated state vector x. To do so I need to find optimal prediction interval n. So, the objective for us now is to find such configuration of state vector x(k) that prediction of y(k) can be done more precisely, i.e. with minimum uncertainty.

Results of calculated in MSFNA number of false neighbors for chosen time lags can be represented in matrix form Fig.2. 1. Multiscale False Neighbors Matrix is defined in such form, that numbers of false neighbors decrease from left corner cell in all directions, this representation of false neighbors gives a clear understanding of number of false neighbors along both radii d_y and d_y.

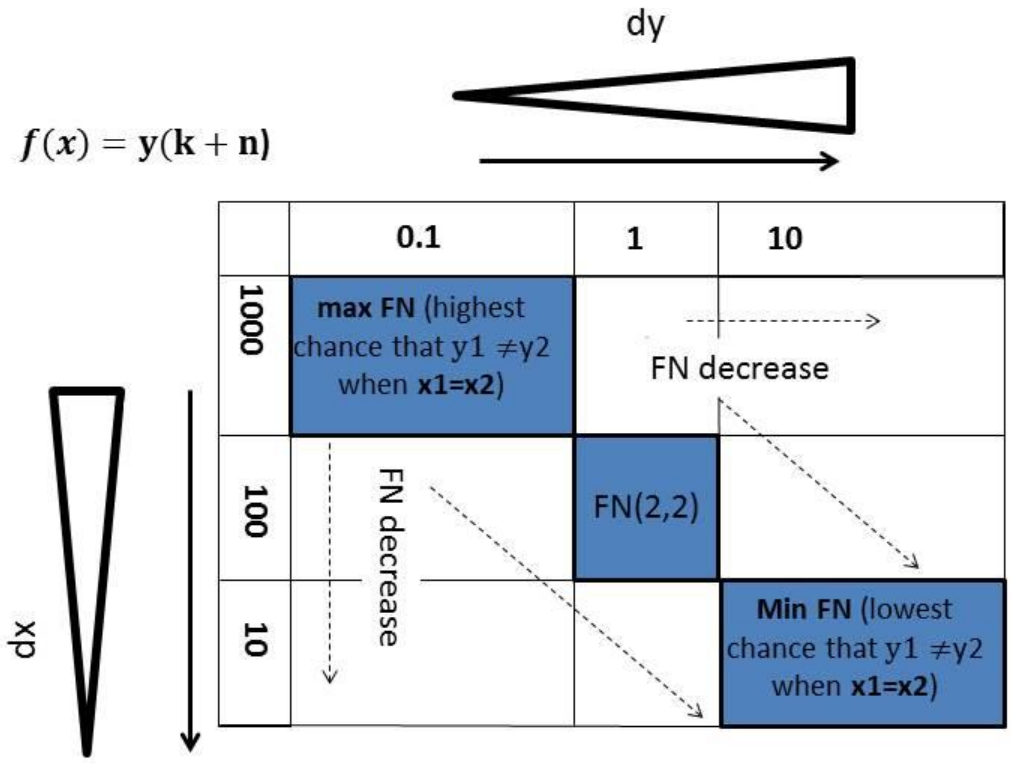


Fig.2. 1: False Neighbors Matrix (FNM). Adopted for this thesis from [1][2][3]

To calculate sum of all FN in MSFN matrix I use Areal Cumulative Count, presented in [1][2][3] as follows:

$$ACFN(\mathrm{n}) = \sum_{i=1}^{k}\sum_{j=1}^{k} FN(\mathrm{i}, \mathrm{j}) \qquad (10)$$

where FN (i, j) is number of false neighbors in cell *[i, j]* and *k* is index of diagonal cell.

For uncertainty evaluation in this work number of ACFN will be compared for every prediction interval *n*. This value will be chosen according to correlation analysis of data.

4 Data Description

4.1 Description of the Theoretical Model of Data

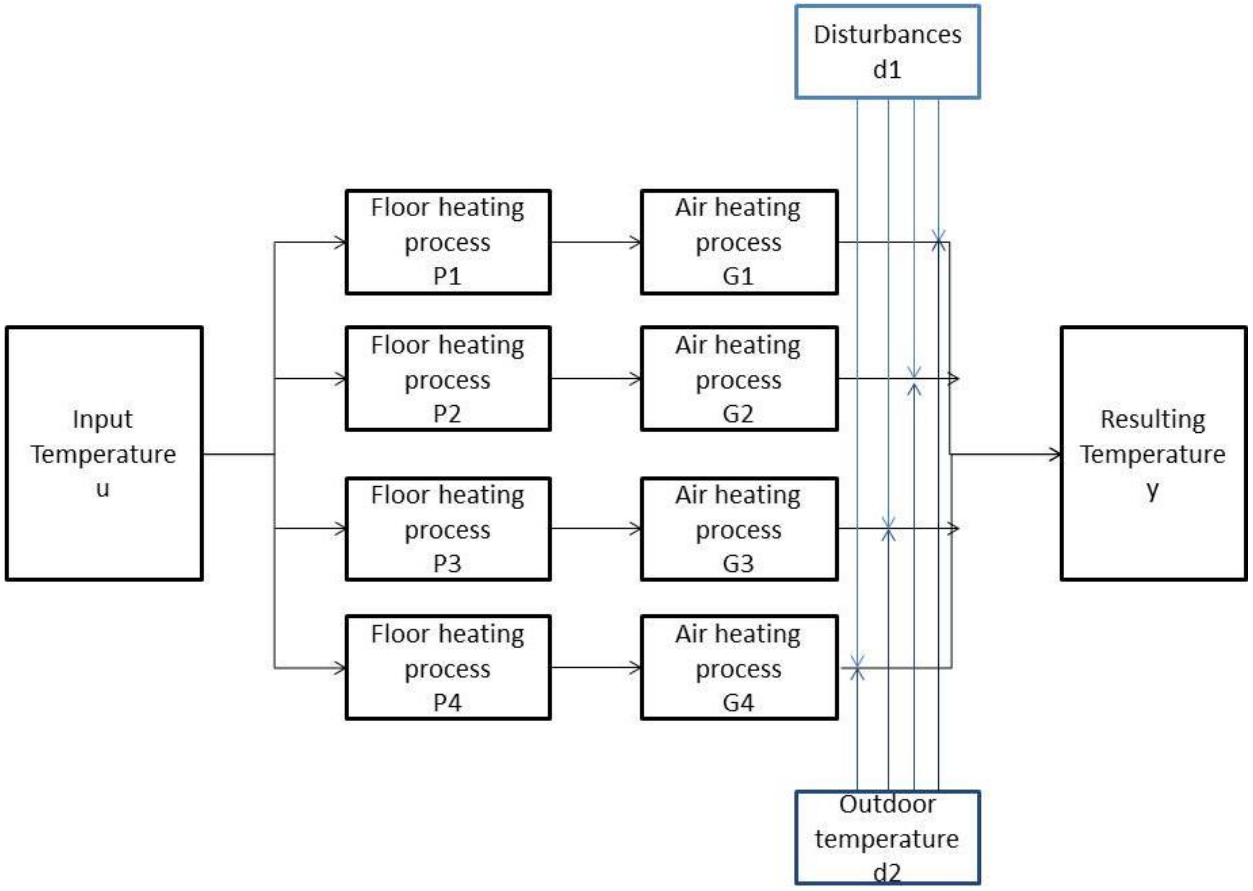


Fig.3. 1: Simplified Theoretical Model of CTU building

Theoretical model of CTU building was simulated in Matlab Simulink. Real CTU building consists of four blocks, each of the blocks is expressed in theoretical model by separate plants.

Incoming hot water, with amplitude 60 (^{o}C) and period 24 hours, starts heating floor of the rooms in every blocks (P1, P2, P3, P4) with randomly chosen time constant (taup) and time delay, then temperature of the air in rooms starts increasing, the process of temperature increasing is represented by subsystems (G1, G2, G3, G4) with randomly chosen time constants (taug). The process is affected by disturbances, such as open doors, windows and number of visitors in a building, and they are expressed by random signal. Outdoor temperature is also considered as disturbance and described by transfer functions. Model has two different time constants that are heating time constant and cooling time constant. These time constants affects model in such a way, that cooling process takes more time than heating process.

Simulation has been made for 900 hours with heating period 24 hours and sampling frequency 6 minutes. Our theoretical model can be considered as causal system, since output is highly dependable on input (see the data in Fig.4.1, p.18)

4.2 Description of the Real Data

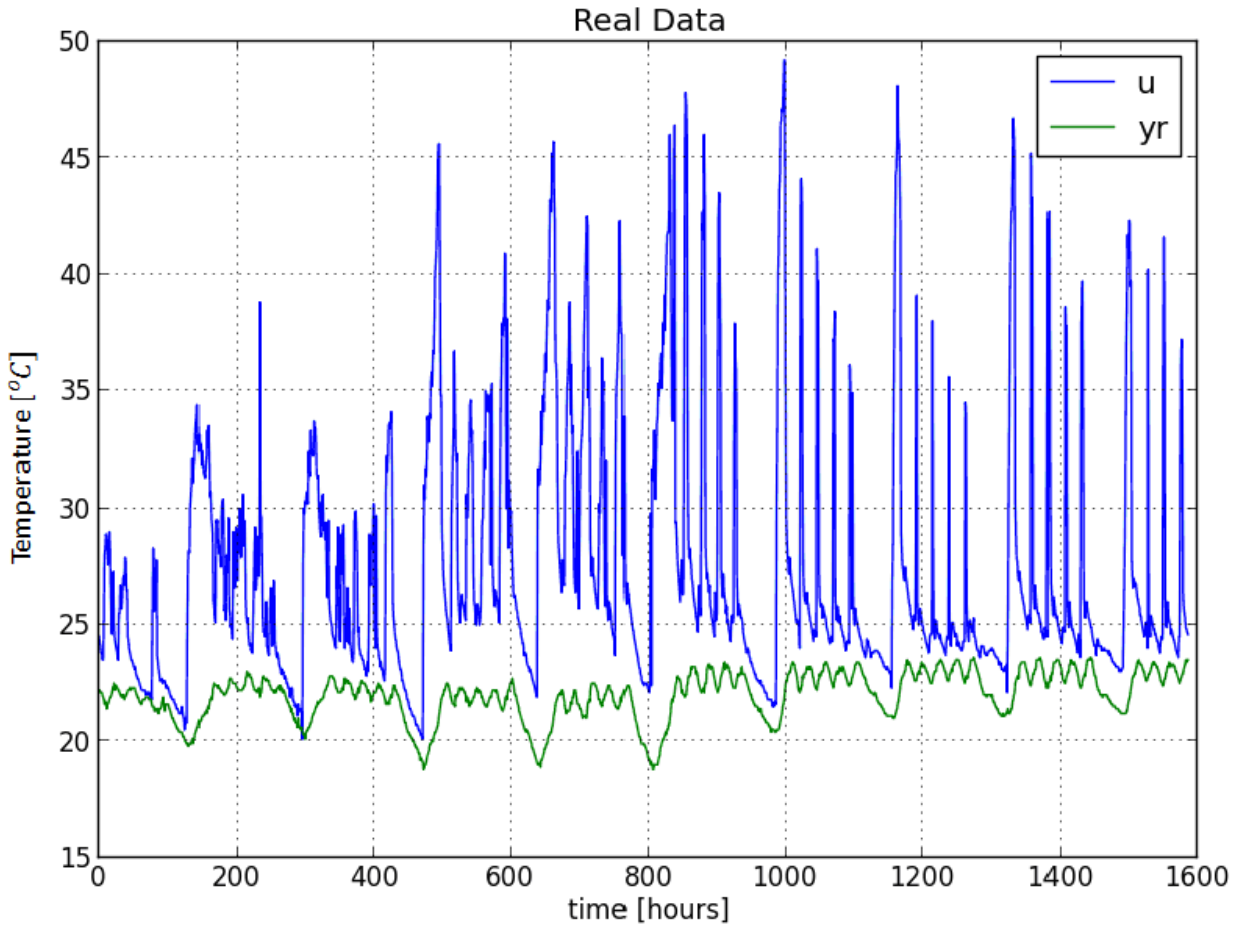


Fig.2. 2: Experimentally measured data from [4]

Real data for this thesis were taken from work [4]. Measurements in [4] were carried out in Czech Technical University (CTU) building during heating season from January to March. CTU building consists of four blocks, the constructions of each blocks are same. It is a five floors building, the heating in building is achieved by supplying mixed hot and cold water. Mixing is made in heat exchanger, which is controlled by PID controller and set point temperature is controlled by high-level controller [4]. Measurements were made hourly for 1600 hours.

I assume that input temperature u is control temperature measured from controller, but it is not set point temperature, therefore not only output temperature depends on input, but also the other way around, since heating system is usually closed loop system. After this assumption I can suppose that system might be a noncausal system

5 Experimental Analysis

In this section all steps of data pre-processing defined in theory above will be performed. First correlation analysis of theoretical data will be outlined, coarse-graining (downsampling and filtering) will be performed along with correlation analysis, then evaluation of uncertainty in data will be made by Multiscale False Neighborhood analysis; same steps will be made for real data pre-processing. In addition, experimental evaluation of minimum uncertainty for different time lags will be presented.

All codes in this work are written in Python programming language, separate code for each step for both theoretical data and real data are carried out. Every step of experimental analysis is provided by graph, explaining and comparing results of data pre-processing step.

5.1 Theoretical Model Data Analysis

First step in data pre-processing is correlation analysis performed by cross-correlation function (2). I calculated cross-correlation function for artificial data on data set; this set is picked out from whole range of data at stable area. Fig.4. 1 is the result of correlation analysis between input temperature u and output temperature y. Cross-correlation analysis shows significant positive correlation between variables r=0.7 at lag 7 hours. Zoomed artificial data in Fig.4. 1 shows that cooling process takes more time than heating, it is because of different time constants of these processes.

Filtering is applied to artificial data in Fig.4. 2. To clearly see the effect of filtering on artificial data I added some noise to it. Filtering is made according to formula (4).

As a next step in data pre-processing I perform Coarse-Graining, represented by downsampling and filtering. Downsampling is calculated using equation (5). Sampling period in artificial data is six minutes. Total number of data is 8991. For CI modeling this amount of data can be too many. So, I need to reduce amount of data, for this purpose downsampling is used. According to formula (5) from the original data we picked out every points with some downsampling rate M. Downsampling for artificial data was made five times with different rates (*M=2, 3, 4, 5*).

After every downsampling filtering and correlation analysis of filtered and downsampled data is performed, to see the effect of downsampling on the system. Fig.4. 5 (a) shows how sampling shifts time lags of maximum correlation coefficient. Highest change in lag is at sampling rate *M=5*, however this change is not very big and this can mean that during downsampling data does not change very much and we do not loose important information. The same conclusion can be made for filtering in Fig.4. 5 (b).

The second objective of this work is to analyze uncertainty in data. This is performed by Multiscale False Neighborhood Analysis. By means of equation (7) number of False Neighbors for sequences of radii is calculated, and FNM is evaluated, then sum of all FN for chosen prediction interval is calculated by Areal cumulative count (10). ACFN of all uncertainties in data helps to find optimal configuration of state vector for Neural Network modeling. Configuration of state vector is chosen according to equation (8).

Our chosen radii are $n_u = 5$ and $n_y = 5$. MSFNA is performed on comparison of choice of different lags n of modeled variable y (9). We use Areal Cumulative count of all FN along both distances d_x and d_y for different time lags n. Time lags n are chosen with respect to cross-

correlation function of artificial data. Program written in Python programming language calculates ACFN for time lags n in range from -12 to 12. Chosen values of distances for d_x are $d_x = 100,\ d_x = 10,\ d_x = 1$. Values of d_y are $d_y = 0.1,\ d_y = 1,\ d_y = 10$.
With regards to chosen variables, and according to equation (8) our state vector will look as follows:

$$\mathrm{x}(k) = \begin{bmatrix} y(k) \\ y(k-1) \\ y(k-2) \\ y(k-3) \\ y(k-4) \\ y(k-5) \\ u(k) \\ u(k-1) \\ u(k-2) \\ u(k-3) \\ u(k-4) \\ u(k-5) \end{bmatrix} \qquad (11)$$

As can be seen from equation (11) state vector for our data is constructed from control input *u(k)* and controlled output *y(k),* where k is discrete index of time with constant sampling.

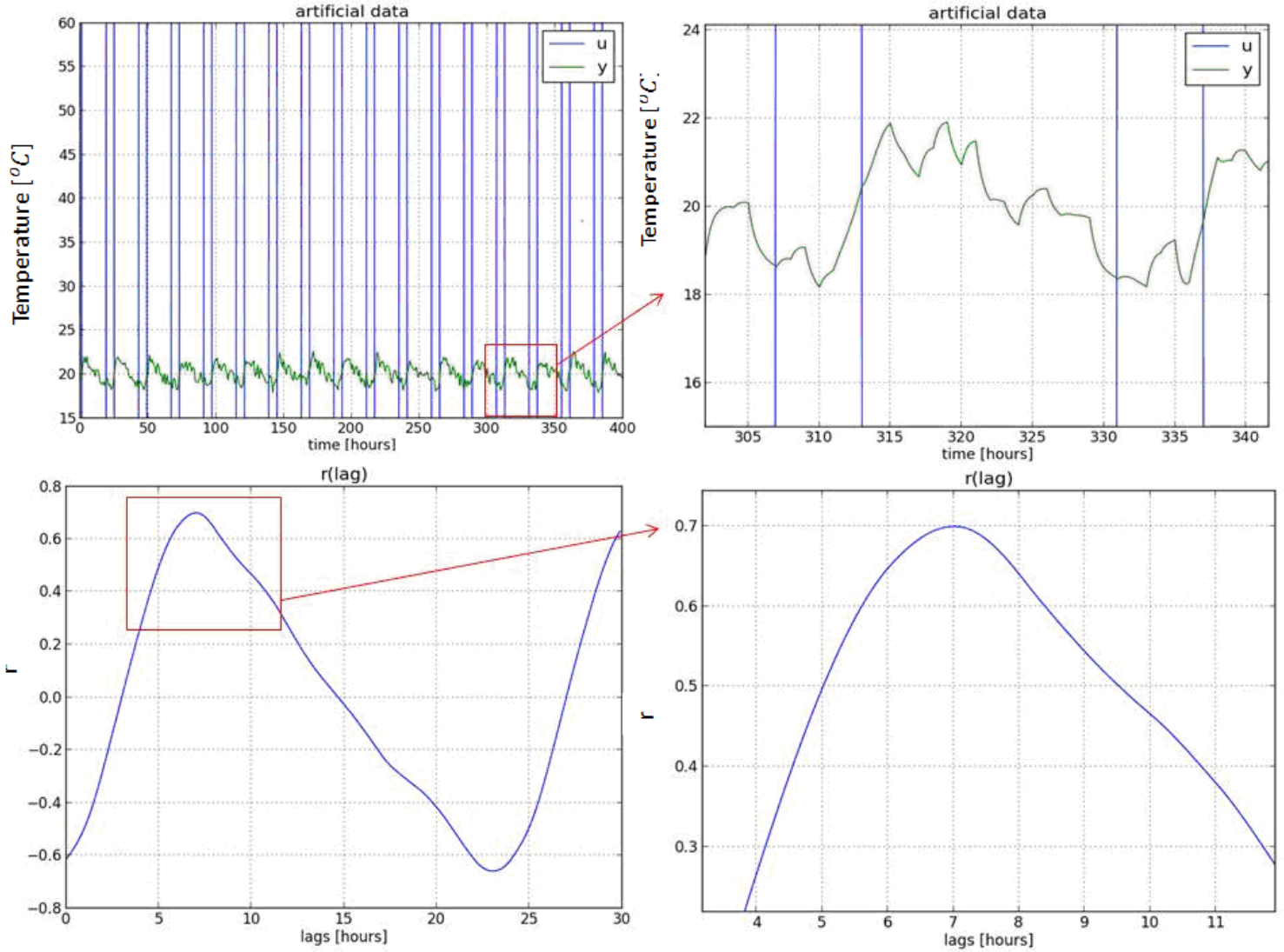


Fig.4. 1: Correlation analysis of artificial data. a) artificial data, b) zoomed artificial data shows time lags in heating process, c) cross-correlation function of input and output temperature, d) zoomed cross-correlation graph shows highest correlation coefficient r=0.7 at time lag 7 hours.

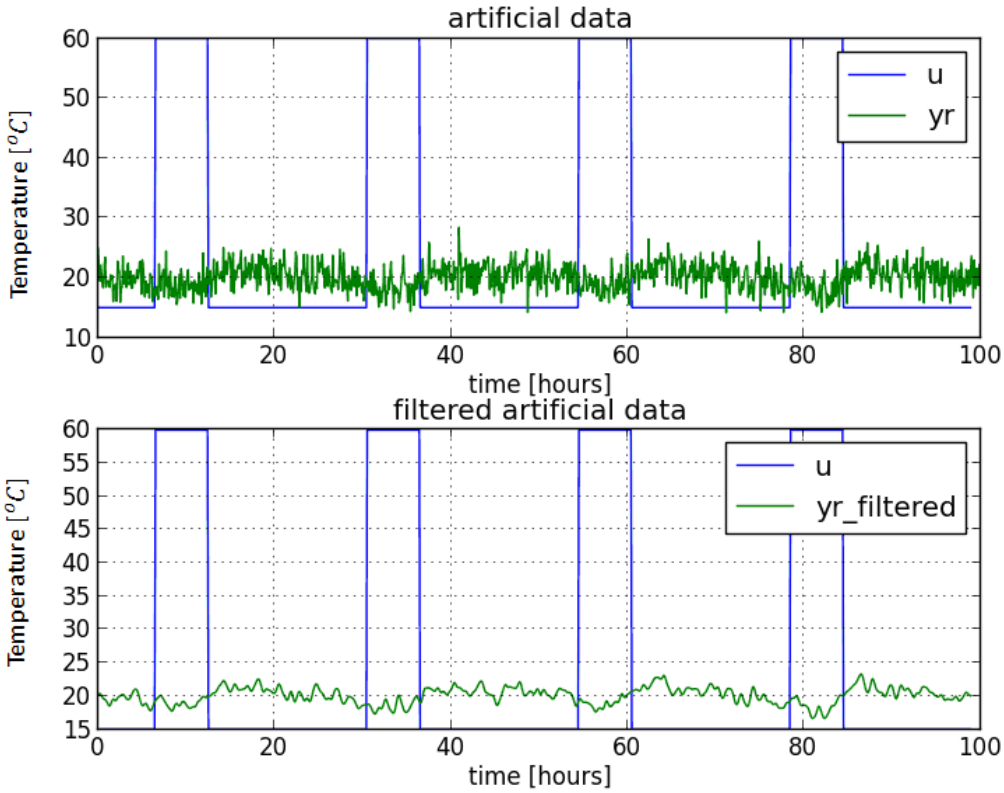


Fig.4. 2: Result of filtering of artificial data. Graphs: a) original artificial data with added noise, b) filtered artificial data.

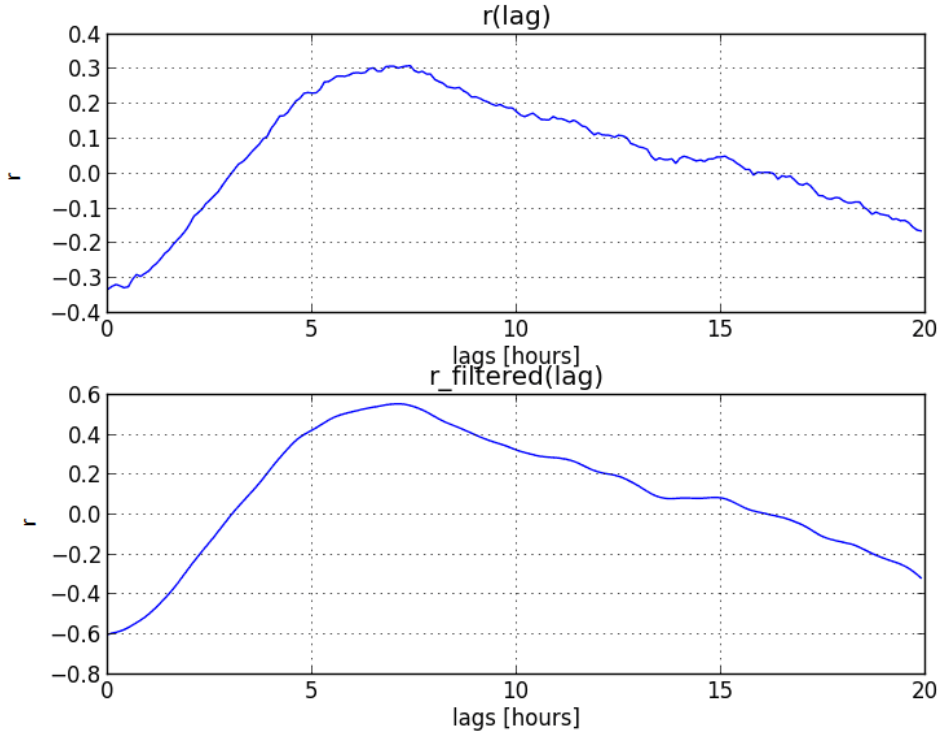


Fig.4. 3: Effect of filtering on cross-correlation function. a) cross-correlation between original artificial data with noise added, maximum correlation coefficient *r=0.4* at lag 6 hours, b) cross-correlation between input temperature u and filtered output temperature y, maximum correlation coefficient *r=0.58* at lag 6 hours

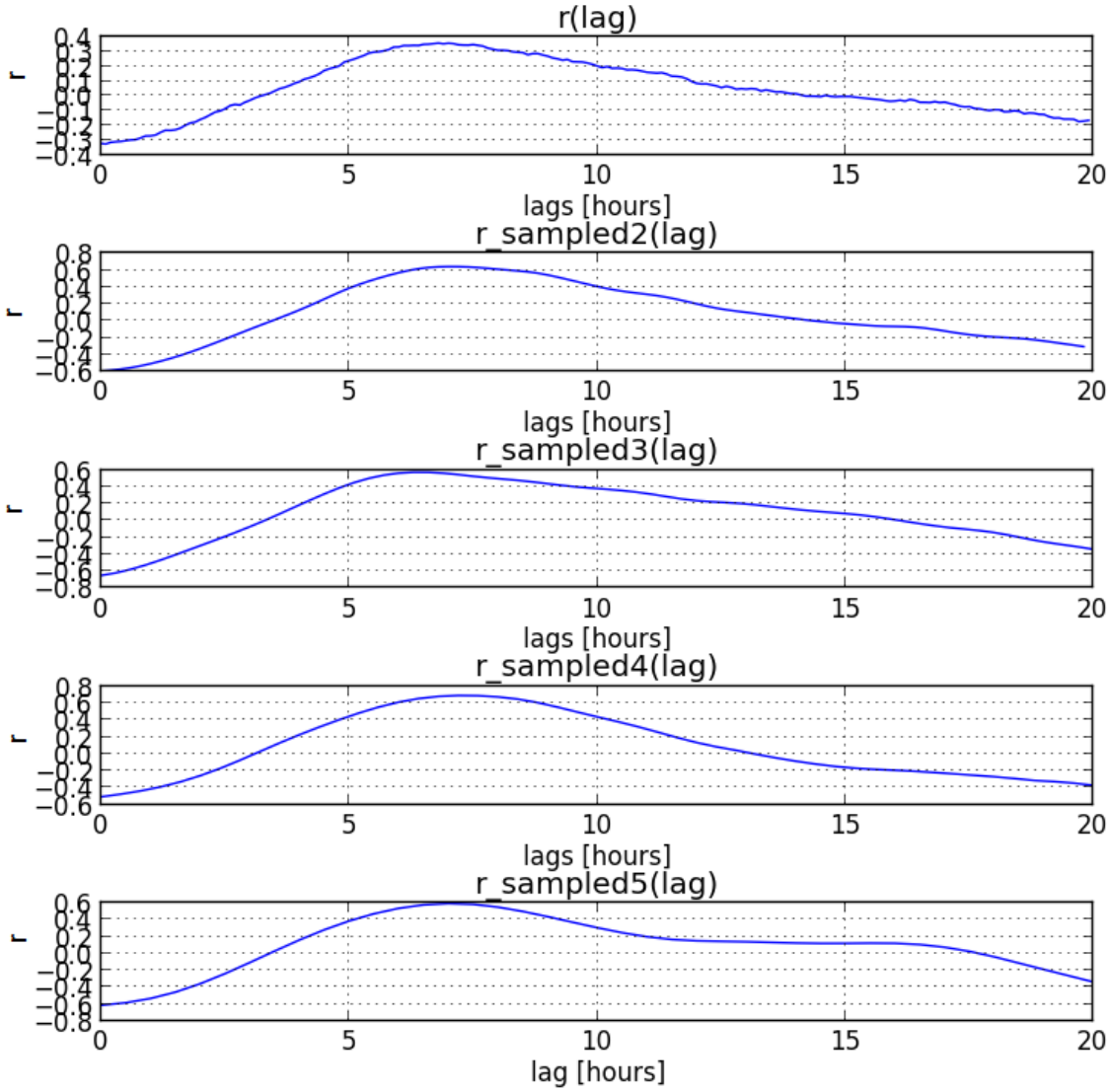


Fig.4. 4: Cross-correlation function of artificial data after sampling and filtering. a) cross-correlation of original data, b) cross-correlation of downsampled with M=2 and filtered data, c)cross -correlation of downsampled with M=3 and filtered data, d) cross- correlation of downsampled with M=4 and filtered data, e) cross-correlation of downsampled with M=5 and filtered data

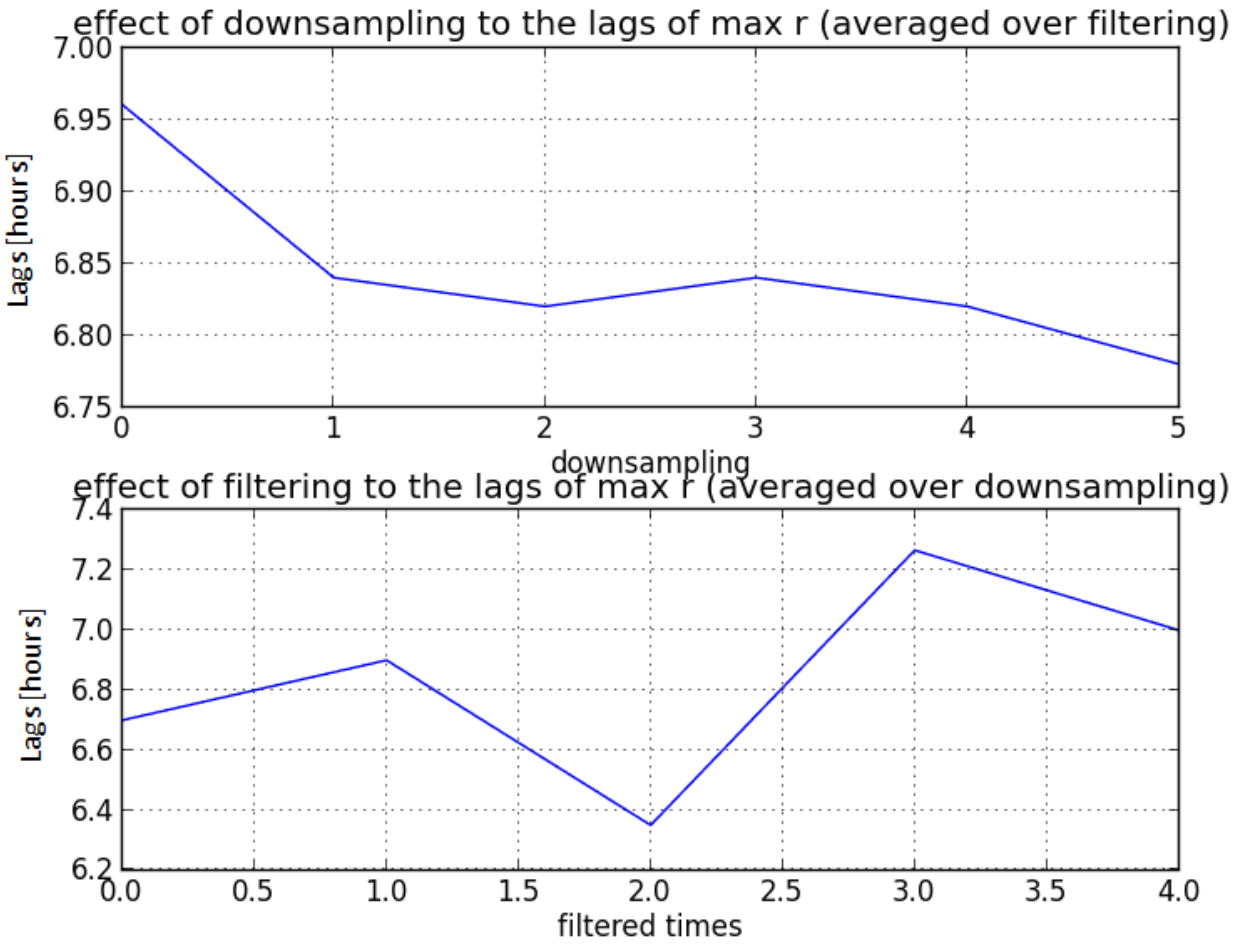


Fig.4. 5: a) Shifting of lags of maximum correlation coefficient after every sampling, b) Shifting of lags of maximum correlation coefficient after filtering of sampled data

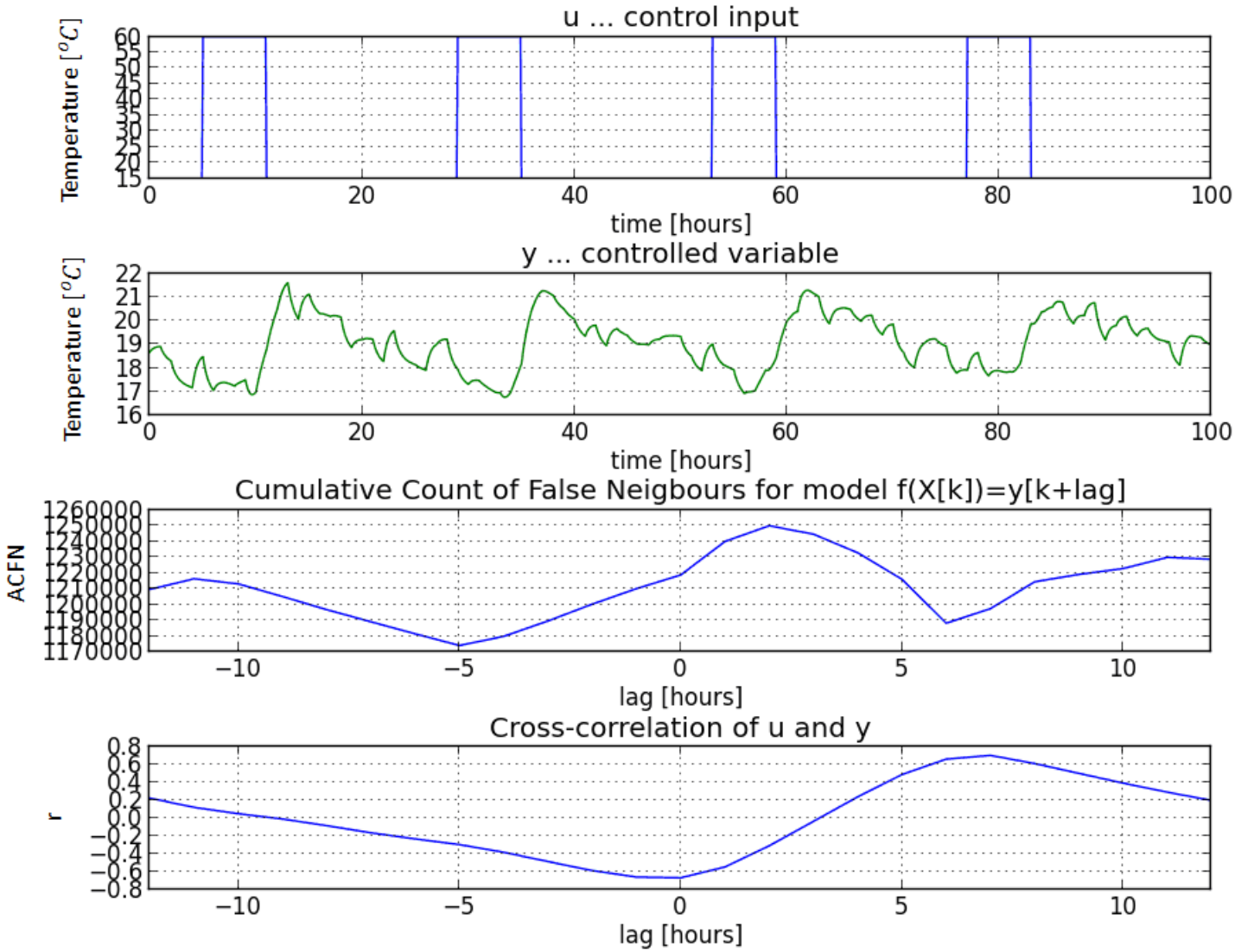


Fig.4. 6: Multiscale False Neighborhood Analysis. a) u control input, b) y controlled variable, c) cumulative count of False Neighbors with smallest numbers of False Neighbors at time lag 6 hours, d) cross-correlation function of input and output temperatures with maximum correlation coefficient at lag 6 hours

5.2 Real Data Analysis

This subsection demonstrates results of experimental analysis on real data. Real data is taken from Reference [4]. Fig.4. 10 shows cross-correlation function of real data. Maximum positive correlation *r=0.4* is indicated at lag 12 hours.

In order to eliminate noise filtering of real data is performed in Fig.4. 8. Resultant output temperature y has been filtered by means of moving average filter (4).

Coarse-graining applied for real data in the same way as for artificial data: I reduce amount of data by means of donwsampling and filter data by moving average filter after every performed downsampling.

Measurement of real data has been made with sampling period one hour. It is an optimal sample rate of measurement, so total amount of data is 1600 measurements. Data does not require downsampling, however this pre-processing step still has been made for real data to understand the effect, but only for two downsampling rates *M=2, M=3* (Fig.4. 10).

The last step in our pre-processing is uncertainty evaluation. Multiscale False Neighbors Analysis is applied for data in range from 100 to 1500 hours. Chosen time lags for configuration of state vector are: $n_u = 10$ and $n_y = 10$. With distances $d_x = 100$, $d_x = 10$, $d_x = 1$ and $d_y = 0.1$, $d_y = 1$, $d_y = 10$. Program for MSFNA written in Python calculates Areal Cumulative Count of all False Neighbors in Multiscale False Neighbors Matrix (Fig.2. 1) for different time lags (prediction intervals) n, chosen according to correlation analysis in range, where maximum correlation coefficient of variables occurs, from -15 to 15 hours.

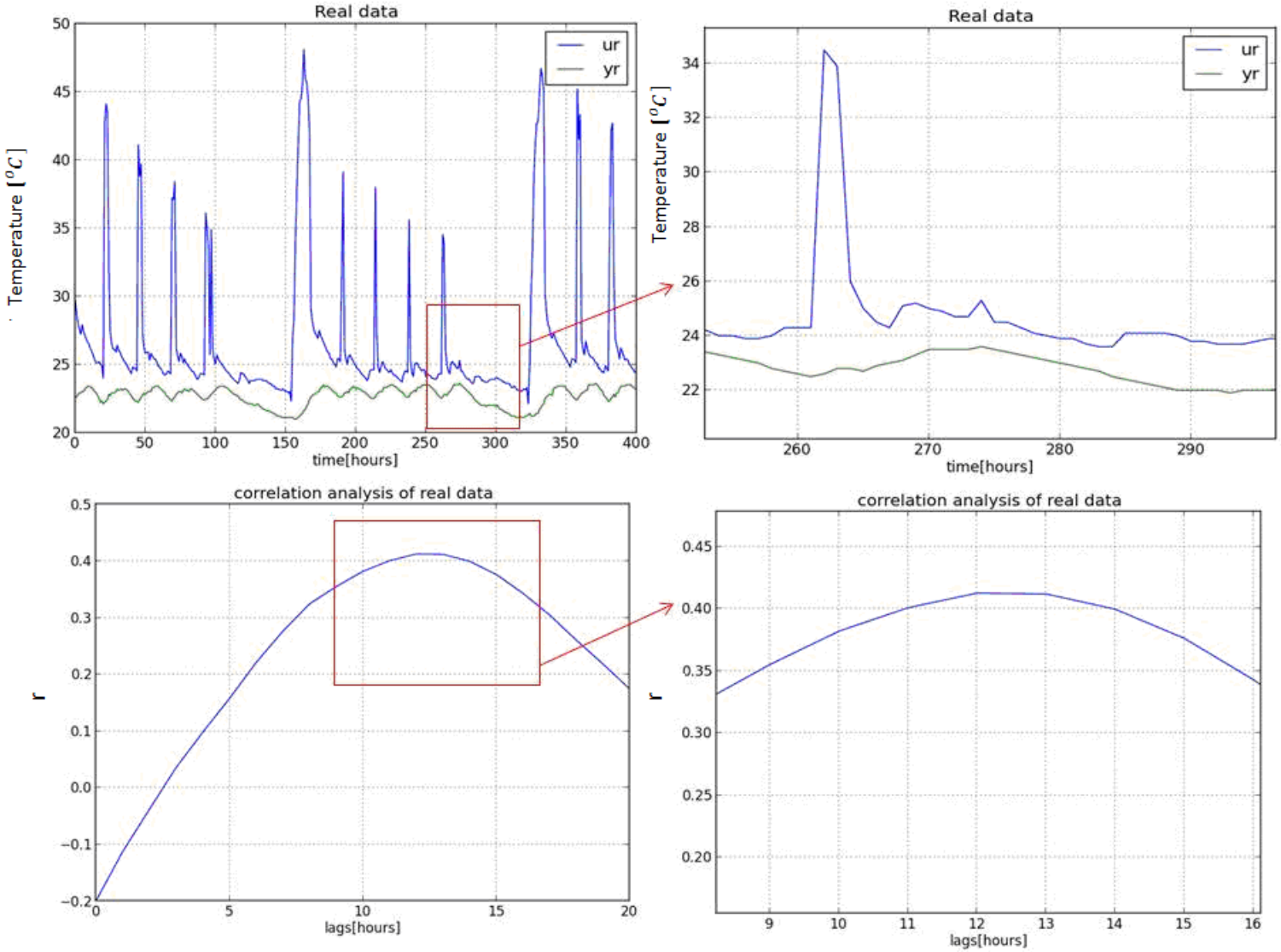


Fig.4.7:Correlation analysis of real data a) chosen stable set of real data, b) zoomed set of real data shows time delays of process, c) cross-correlation function of input temperature and measured temperature of the building, d) zoomed cross-correlation function at maximum correlation coefficient r=0.412 at lag 12 hours

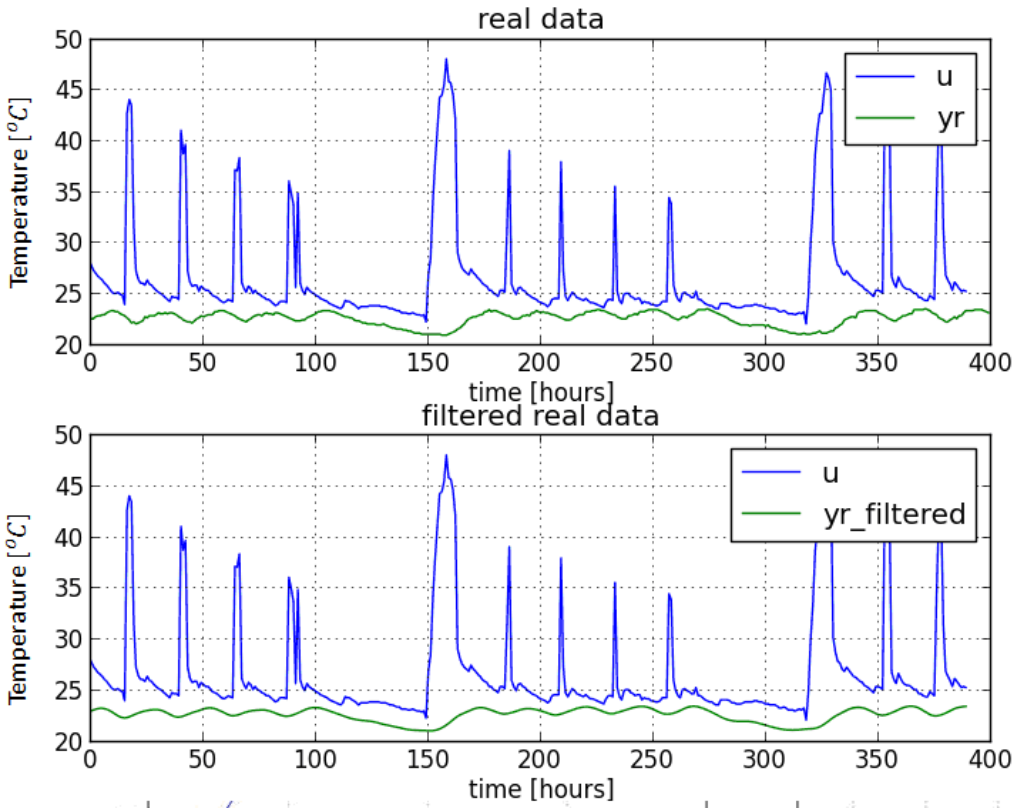


Fig.4. 8: Result of filtering of real data. Graphs: a) original real data, b) filtered real data.

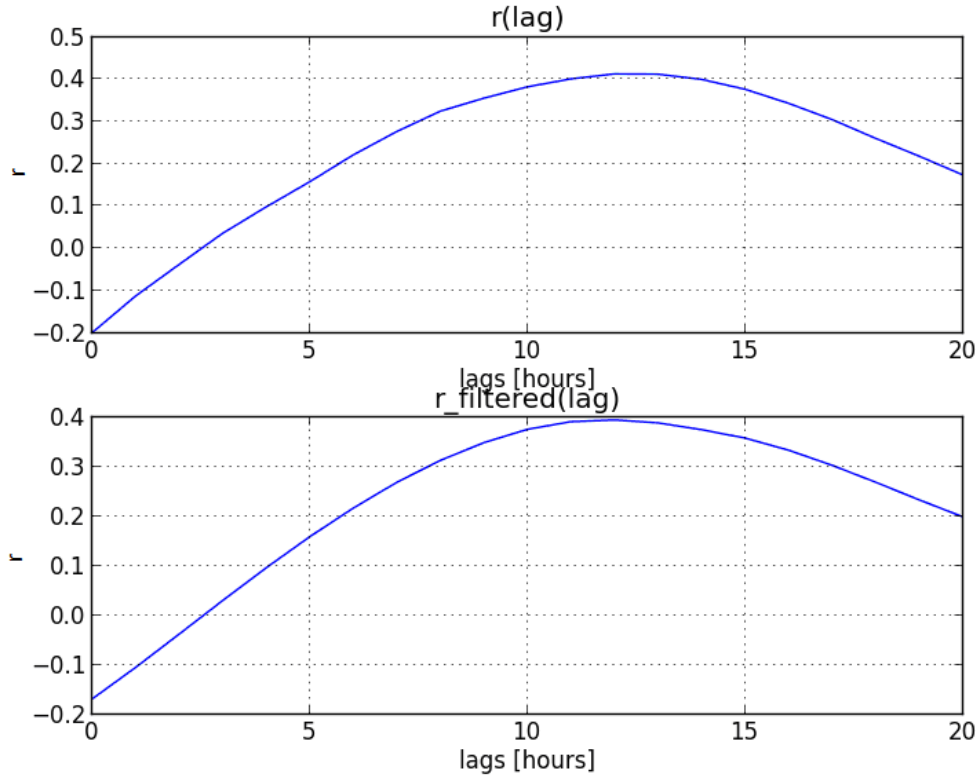


Fig.4. 9: Effect of filtering on real data cross-correlation function, a) cross-correlation of original data, b)cross-correlation of filtered data

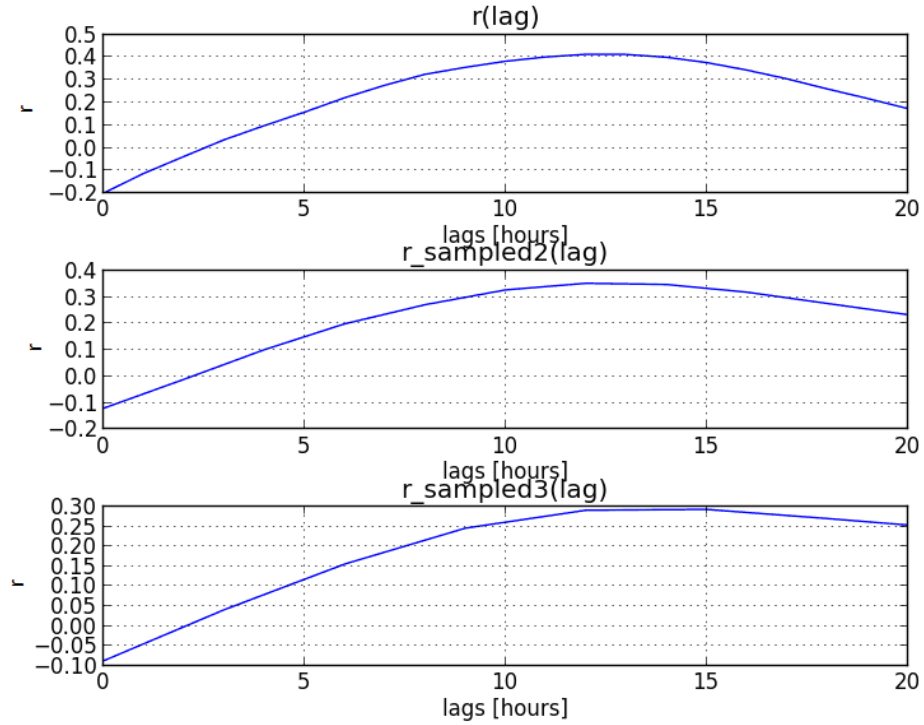


Fig.4. 10: Cross-correlation function of real data after coarse-graining. a) cross-correlation of original data, b) cross-correlation of downsampled with M=2 and filtered data, c) cross-correlation of downsampled with M=3 and filtered data

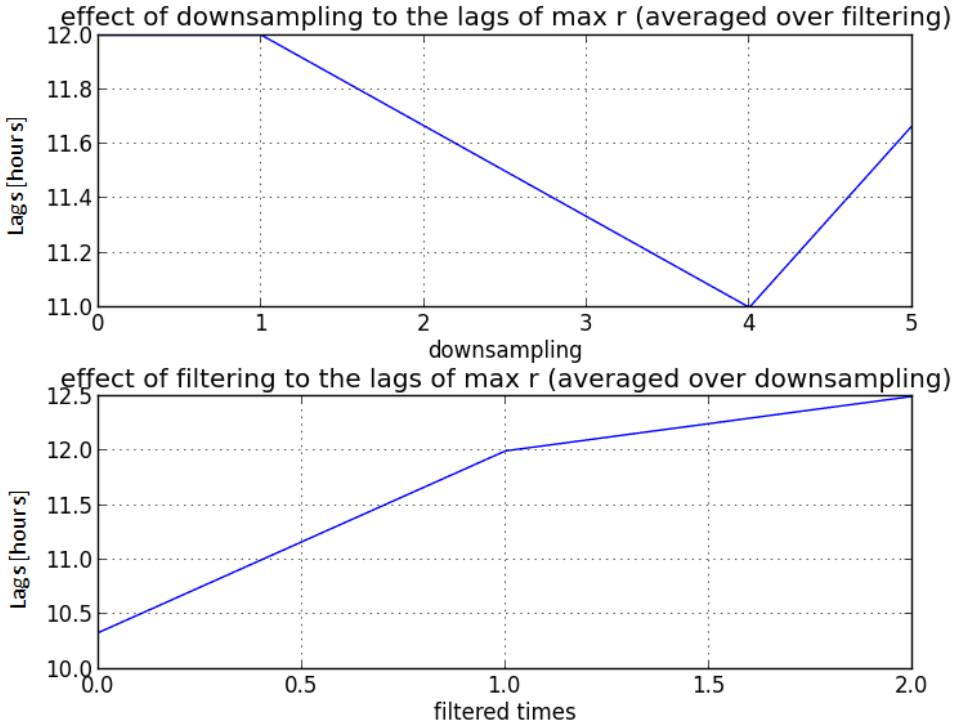


Fig.4. 11: a) Shifting of lags of maximum correlation coefficient after every downsampling, b) Shifting of lags of maximum correlation coefficient after filtering of sampled data

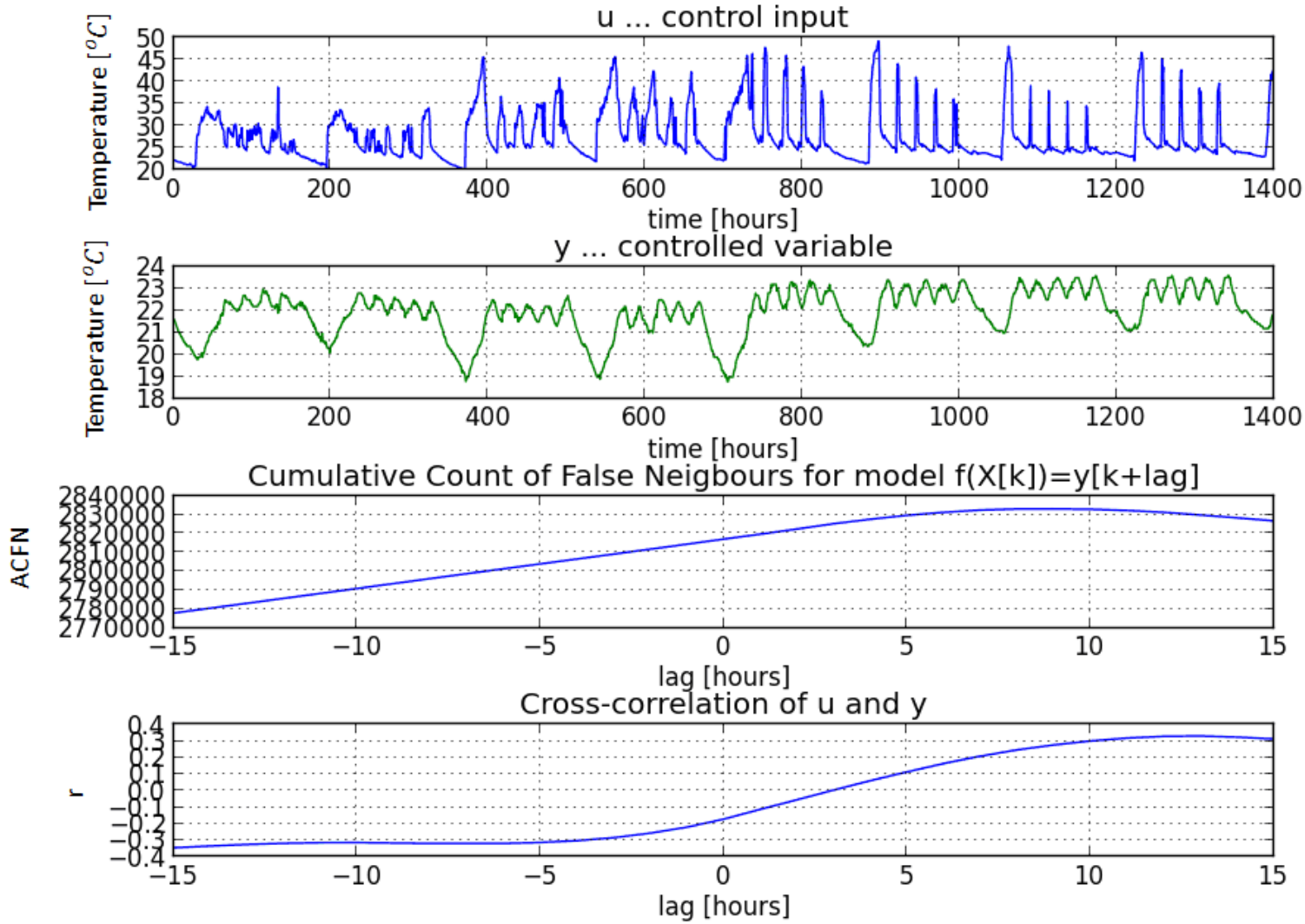


Fig.4. 12: Multiscale False Neighborhood Analysis of real data. a) u control input, b) y controlled variable, c) cumulative count of False Neighbors with smallest numbers of False Neighbors, d) cross-correlation function of input and output temperatures with maximum correlation coefficient at lag 12 hours

6 Discussion

As stated before, objectives of this work were to perform data pre-processing for both theoretical model data and real data and to find best prediction interval for chosen configuration of state vector for possible further Neural Network modeling. In this section I describe in detail results of each performed pre-processing step of both artificial and real data in the same order, as experimental analysis was carried out.

First applied step in experimental analysis is correlation analysis. Fig.4. 1 and Fig.4. 8 are resultant pictures of analysis of artificial data and real data respectively. In case of correlation analysis of artificial data cross-correlation function shows significant positive correlation between variables $r=0.7$ at lag 7 hours. This time lag identifies time delay of the system. According to position of maximum correlation coefficient, I assume that best prediction interval for theoretical model data is lag $n=7$ hours for equation (9).

Cross-correlation function of real data is shown in Fig.4. 10, indicating maximum positive correlation $r=0.4$ at lag 12 hours. Dynamic of the system is complex; from zoomed figure it is clearly seen, that process has two different time delays. One of them is time delay between time when input temperature u starts increasing and time when temperature y responses to this change; this delay is equal to one hour. The other delay is between input temperature dropping down and output temperature starting to respond, this delay is 12 hours. It is interesting that cooling time delay is equal to time lag of maximum correlation coefficient. From performed analysis I suggest that best prediction can be at made lag 12 hours.

From Fig.4. 1 and Fig.4.7 difference between real model data and theoretical model data can be seen. As mentioned before, theoretical model is constructed in such a way, that it has two time constants one of heating process, second of cooling. In case of real data, both processes have approximately same time.

The next steps in data pre-processing is correlation analysis with coarse-graining technique. This technique as described above is divided in two steps: filtering and downsampling with filtering. The effect of filtering on artificial data and on cross-correlation function of the system can be seen from Fig 4. 2 and Fig.4. 3 respectively. Some noise is added to artificial data to clearly see the effect filtering can provide us with. Filtered data, besides smoothing and denoising of controlled variable u, improved cross-correlation function, without shifting it. Maximum correlation coefficient increased from $r=0.4$ to $r=0.7$ at the same time lag, this can be considered as a big advantage, since $r=0.7$ indicates significantly strong positive correlation between input and output temperatures.

Filtering of real data has not showed as good result, as in case of artificial data. However, filtering provided us with smoother signal, with eliminated noise, but without any changes in correlation analysis (Fig.4. 8).

Second part of coarse-graining is downsampling. Downsampling is performed together with filtering, with presumption that it would have positive influence on data. Goarse-graining presented in such way increases value of maximum correlation coefficient. For artificial data filtering and downsampling do not make big shifting of time lags of maximum correlation coefficient Fig.4. 5. and Fig.4.4.

In case of real data coarse-graining does not make any improvements, moreover in case of downsampling with rate M=3 it makes correlation between variables weaker and shifts maximum correlation value for one hour.

Uncertainties that can be contained in data may affect CI modeling of system not in a good way. Smaller number of uncertainties in data can help to achieve good and reliable result from CI modeling. MSFNA is used in this thesis to find prediction interval n for modeled variable (9) with minimum number of uncertainties for a chosen state vector (8). Fig.4. 6 represents MSFNA for artificial data. Graph shows smallest cumulative count (10) for positive prediction interval $n = 6$ hours that absolutely corresponds to cross-correlation analysis performed above. However, minimum number of FN is at time lag -5, but this lag cannot be considered because prediction makes sense only for positive time lag. So, for state vector with configuration (11), best prediction interval for Neural Network modeling is $n = 6$:

$$\mathbf{y}(k+6) = \mathrm{f}(\mathbf{x}(k))$$

MSFNA for real data is calculated for configuration of state vector (8). Fig.4. 12 shows that MSFNA has revealed uncertainty in input-output mapping for real data. This statement is made because for time lag 12 cumulative count of FN is highest, that is in contradiction with correlation analysis and does not allow us find prediction interval. However, there is a decrease of ACFN in negative area of prediction lag. In addition, it coincides with absolute value of maximum correlation coefficient in negative time lags area. To understand reason caused the failure of analysis more information on controller and measuring way is needed.

The cross-correlation function implies that one of the correlated variables depends on another, in such a way that one leads another. In case of temperature, I assume that input temperature leads output temperature, so output depends on input temperature, and system can be considered as causal system. It is a relationship that is in our theoretical data, since theoretical model does not use any controller or feedback control, control input u is same as temperature set point Fig.3. 1. In case of real data, there is strong dependence on negative lags area, such a system may be considered as non-causal, because this relationship states that control input may be dependent on controlled variable. PID controller used as low-level control and closed loop control of building heating system may be the reason of different time delays of system. I can suppose, that temperature used as control input is actually control variable from controller and it depends on controlled variable y. Thus, the assumption that system is not causal in sense of y=f(x) seriously complicates the training of Neural Network models.

7 Conclusions

This work is dedicated to data pre-processing of heating system for computational intelligence modeling. Several simple methods of pre-processing were introduced and programmed for theoretical model data as well as for experimentally measured data. Data pre-processing steps were graphically represented and their effects to original data were compared.

In this work I adopted data pre-processing technique based on correlation analysis. I identified relationship between variables and used correlation to evaluate best prediction interval in correspondence to the uncertainty to input-output data.

I introduced and investigated a unique utilization of coarse-graining as for filtering and sampling method. Filtering with moving average filter showed relatively good result in noise reduction and smoothing data. Downsampling and filtering applied when resampling of large amount of data was needed. The advantage of donwnsampling in case of artificial data was that it reduced number of data, without significant change in time lags of maximum correlation coefficient, and even increased value of maximum correlation coefficient of variables. While filtering of real data improved signal and partially eliminated noise, downsampling did not show any significant result, and moreover it shifted maximum value of correlation coefficient for an hour that can be considered as disadvantage Fig.4. 11.

The third objective of this work was to provide uncertainty evaluation in data by Multiscale False Neighborhood Analysis (MFSNA). I compared cumulative sum of all uncertainty in data for various different prediction intervals. It was important for us was to see if prediction interval when evaluated by MSFNA corresponds to lags of maximum correlation coefficient.

Noisy yet ideal artificial data showed result that ideally corresponded to lags of correlation analysis Fig.4. 6, we found before. Therefore, we can conclude that prediction time=6 hours is for given input data most optimal.

MSFNA for real data, i.e., for data measured inside a control loop, showed different relationship between correlation and uncertainty in data. Minimum of false neighbors did not correspond to maximum cross-correlation; moreover cumulative sum had maximum value at prediction time assumed by correlation analysis.

Cross-correlation besides evaluating relationship among variables helped us to assume prediction time for further CI modeling. It was important, that I could approach causality evaluation of data from system via cross-correlation and uncertainty; the analyses studied in this thesis suggest that correlation analysis shall be accompanied by analysis of uncertainty for better understanding data configuration for neural network models.

Thus, this thesis is also a very first thesis on novel evaluation of appropriateness of prediction horizon for neural networks by correlation analysis together with uncertainty analysis by multiscale false neighborhood analysis.

References

[1] Bukovsky, I., Kinsner, W., Maly, V., Krehlik, K.: "Multiscale Analysis of False Neighbors for State Space Reconstruction of Complicated Systems", in proceedings of *2011 IEEE Symposium Series on Computational Inteligence (SSCI), IEEE Workshop CompSens 2011,* Paris 2011, pp. 35-41 ISBN 978-1-4577-0470-3.

[2] Bukovsky, I., Kinsner, W., Bila, J.: "Multiscale Approach to Uncertainty Evaluation of Input-Output Data", *Automatizácia a riadenie v teórii a praxi ARTEP 2012, Slovakia 2012, ISBN*: 978-80-553-0835-7.

[3] Bukovský, I.: Multiscale Analysis Approach: "Trend in Evaluation of Measurements and in Signal Processing", In Sborník odborného semináře *Nové metody a postupy v oblasti přístrojové techniky, automatického řízení a informatiky* 2011. Praha: Ústav přístrojové a řídicí techniky FS ČVUT, 2011, p. 11-16. ISBN 978-80-01-05041-5.

[4] Siroky J., Oldewurtel F., Cigler J., Privara S.: "Experimental analysis of model predictive control for an energy efficient building heating system", Applied Energy 88 (2011) 3079-3087, 2011.

[5] Mirinejad H., Sadati S.H., Ghasemian M., Torab H., "Control Techniques in Heating, Ventilating and Air Conditioning (HVAC) Systems", Journal of Computer Science 4(9): 777-783, 2008. ISSN 1549-3636.

[6] Smith S. W.: "The Scientist's and Engineer's Guide to Digital Signal processing", California Technical Publishing 1997-1999, p. 277-284.

[7] Madsen H.:"Time series analysis", Technical University in Denmark, 2008, pp.78-80, pp.248-290, ISBN-13:978-1-4200-5967-0

[8] Villar J.R, Enrique de al Cal, Sedano J.:"A Fuzzy Logic based efficient energy saving approach for domestic heating systems", Integrated Computer-Aided Engineering 15 (2008) 1-9, Spain, 2008, ISSN 1069-2509/08

[9] Yang J., Rivard H., Zmereanu R.: "Building Energy prediction with Adaptive Artificial Neural Network", Building Simulation 2005, *Ninth International IBPSA Conference,* Montreal Canada, August 15-18, 2005.

[10] Moon J.W., Jung S.K., Kim J.:"Application of ANN (Artificial Neural Network) in Residential Thermal Control", *Building Simulation 2009, Eleventh International IBPSA Conference,* Glasgow Scotland, July 27-30, 2009.

[11] Rezaee A., Khalil Copayegani M.: "Intelligent Control of Cooling-Heating Systems by Using Emotional Learning", *Electronics and Electrical Engineering 2012 No.4(120),* Alborz Iran, ISSN 1392-1215.

[12] Ben-Nakhni A.E., Mahmoud M.A.: "Energy conservation in building through efficient A/C control using neural network", *Applied Energy 73 (2002) 5-23, 2002,* Department of Mechanical Engineering, College of Technological Studies, Kuwait, Rumaithya 25562, Kuwait, S0306-2619(02)0027-2.

[13] Kennel M.B., Brown R., Abarbanel H.D.I.: "Determining embedding dimension for phase-space reconstruction using a geometrical construction ", Physical Review A Volume 45, Number 6, California, 15 March 1992

[14] Yu L., Wang S., Lai K.K.: "An integrated Data Preparation Scheme for Neural Network Data Analysis", *IEEE Transactions on Knowledge and Data Engineering, Vol.18 No.2,* China 2006, 1041-4347/06

[15] Bauer M., Scartezzini J.-L.: “A simplified correlation method accounting for heating and cooling loads in energy-efficient buildings”, Energy and Buildings 27 (1998) 147-154, Lausanne, Switzerland 16 June 1997, 0378-7788/98

[16] Sliskovic D., Grbic R., Hocenski Z.:“Online Data Preprocessing in the adaptive process model building based on plant data”, Technical Gazette 18, 1(2011), 41-50, ISSN 1330-3651

[17] Bang Y.K., Lee C.H.:”Fuzzi Time Series Prediction with Data Preprocessing and Error Compensation Based on Correlation Analysis”, Third 2008 International Conference on Convergence and Hybrid Information Technology, Kangwondo Korea, 2008, 978-07695-3407-7/08

Appendix

Correlation_analysis_art_data.py

```
from numpy import *
from matplotlib.pyplot import *
import matplotlib

u=loadtxt('u.txt')[3000:7000]
y=loadtxt('y.txt')[3000:7000]


N=len(u)
dt=6 #minutes
t=arange(len(y))*dt/60.
figure()
plot(t,u, label='u')
plot(t,y,'g', label='y'), xlabel('time [hours]')
legend(loc='upper right', numpoints = 1)
title('artificial data')
grid()

##----cross-correlation function------
lags=arange(0,300)
N=len(u)
r=zeros(len(lags))
for lag in lags:
    yshift=y[lag:N]
    ushift=u[0:N-lag]
    r[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
    NN=len(ushift)

lih=lags*dt/60. #lags in hours
figure()
plot(lih,r, label='r' ), xlabel('lags [hours]')
title('r(lag)')
grid()
show()
```

Correlation_analysis_real_data.py

```
from numpy import *
from matplotlib.pyplot import *
import matplotlib

data=loadtxt('data_whole_year.txt')

u=data[1000:1400,1]   # hot water
y=data[1000:1400,0]   # bulding (rooms average) temp

N=len(u)

figure()
plot(u, label='ur')
plot(y,'g', label='yr'),xlabel('time[hours]')
legend(loc='upper right')
title('Real data')
grid()
```

```
lags=range(0,25)
N=len(u)
r=zeros(len(lags))

for lag in lags:
    yshift=y[lag:N]
    ushift=u[0:N-lag]
    r[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
    NN=len(ushift)
```

Correlation_analysis_art_data_filtering.py

```
from numpy import *
from numpy.random import randn
from matplotlib.pyplot import *
import matplotlib


u=loadtxt('u.txt')[6000:7000]
yr=loadtxt('y.txt')[6000:7000]
N=len(yr)
noise=loadtxt('noise.txt')
yr=yr+noise

N=len(u)

lags=arange(0,200)
N=len(u)
r=zeros(len(lags))

for lag in lags:
    yshift=yr[lag:N]
    ushift=u[0:N-lag]
    r[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
    NN=len(ushift)
#--------filtering signal-------------
yrf=zeros(N)
for k in range(1, N-1):
    yrf[k]=1./3*(yr[k-1]+yr[k]+yr[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
```

```
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])

rf=zeros(len(lags))
for lag in lags:
    yshift=yrf[lag:N]
    ushift=u[0:N-lag]
    rf[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

dt=6 #sampling rate
t=arange(len(u))*dt/60.
t2=arange(len(yrf))*dt/60.
lih=lags*dt/60.
figure()
subplot(211)
plot(lih,r, label='r' ),xlabel('lags [hours]')
title('r(lag)')
grid()
subplot(212)
plot(lih,rf, label='rf' )
title('r_filtered(lag)'),xlabel('lags [hours]')
grid()
figure()
subplot(211)
plot(t,u, label='u')
t=arange(len(yr))*dt/60.
plot(t,yr,'g', label='yr'),xlabel('time [hours]')
legend(loc='upper right', numpoints = 1)
title('artificial data')
grid()
subplot(212)
plot(t2,u, label='u')
plot(t2,yrf,'g', label='yr_filtered'), xlabel('time [hours]')
legend(loc='upper right', numpoints = 1)
title('filtered artificial data')
grid()

figure()
plot(u, label='u')
plot(yrf,'g', label='yr_filtered'), xlabel('time [hours]')
legend(loc='upper right', numpoints = 1)
title('filtered artificial data')
grid()

show()
```

Correlation_analysis_real_data_filtering.py

```
data=loadtxt('data_whole_year.txt')

u=data[1000:1400,1]    # hot water
yr=data[1000:1400,0]    # bulding (rooms average) temp

N=len(u)

lags=arange(0,24)
N=len(u)
r=zeros(len(lags))

for lag in lags:
    yshift=yr[lag:N]
    ushift=u[0:N-lag]
    r[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
    NN=len(ushift)
#--------filtering signal-------------
yrf=zeros(N)
for k in range(1, N-1):
    yrf[k]=1./3*(yr[k-1]+yr[k]+yr[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
u=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
#---------------------------------------
rf=zeros(len(lags))
for lag in lags:
    yshift=yrf[lag:N]
    ushift=u[0:N-lag]
    rf[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
```

Correlation_analysis_art_data_coarse_graining.py

```
u=loadtxt('u.txt')[6000:7000]
yr=loadtxt('y.txt')[6000:7000]
N=len(yr)
noise=loadtxt('noise.txt')
yr=yr+noise

#================================
# ORIGINAL SAMPLIG PERIOD dT:
dT=6  #[min]
#================================
lags=arange(0,200)
N=len(yr)
r=zeros(len(lags))
yrf=zeros(N)

#------filtering  signal-------
for lag in lags:
    yshift=yr[lag:N]
    ushift=u[0:N-lag]
    r[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrf[k]=1./3*(yr[k-1]+yr[k]+yr[k+1])

rcg1=zeros(len(lags))
uf=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for lag in lags:
    yshift=yrf[lag:N]
    ushift=uf[0:N-lag]
    rcg1[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
rcg2=zeros(len(lags))
uf=u[1:N-1]
yrf=yrf[1:N-1]
N=len(uf)

for lag in lags:
    yshift=yrf[lag:N]
    ushift=uf[0:N-lag]
    rcg2[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
rcg3=zeros(len(lags))
uf=u[1:N-1]
yrf=yrf[1:N-1]
N=len(uf)

for lag in lags:
    yshift=yrf[lag:N]
    ushift=uf[0:N-lag]
    rcg3[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
```

```
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
rcg4=zeros(len(lags))
uf=u[1:N-1]
yrf=yrf[1:N-1]
N=len(uf)
for lag in lags:
    yshift=yrf[lag:N]
    ushift=uf[0:N-lag]
    rcg4[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
#-------end of filtering----------
#-------Cross-correlation forfiltered data--------
rcg5=zeros(len(lags))
uf=u[1:N-1]
yrf=yrf[1:N-1]
N=len(uf)
for lag in lags:
    yshift=yrf[lag:N]
    ushift=uf[0:N-lag]
    rcg5[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

##---------sampling and filtering-----------
###---------downsampling with rate M=2----------------------
lags2=arange(0,100)
yrsf=yr[::2].copy()
ursf=u[::2].copy()
N=len(yrsf)
figure()
plot(yrsf,'g')
plot(ursf)

rsf=zeros(len(lags2))
for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf[k]=1./3*(yrsf[k-1]+yrsf[k]+yrsf[k+1])
rsf1=zeros(len(lags2))
ursf=ursf[1:N-1]
yrsf=yrsf[1:N-1]
N=len(yrsf)
```

```
for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf1[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf[k]=1./3*(yrsf[k-1]+yrsf[k]+yrsf[k+1])
rsf2=zeros(len(lags2))

for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf2[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf[k]=1./3*(yrsf[k-1]+yrsf[k]+yrsf[k+1])
rsf3=zeros(len(lags2))

for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf3[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf[k]=1./3*(yrsf[k-1]+yrsf[k]+yrsf[k+1])
rsf4=zeros(len(lags2))

for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf4[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf[k]=1./3*(yrsf[k-1]+yrsf[k]+yrsf[k+1])
rsf5=zeros(len(lags2))
###----------Downsampling with M=3---------
lags3=arange(0,80)
yrsf3=yr[::3].copy()
ursf3=u[::3].copy()
N=len(yrsf3)
figure()
plot(yrsf3, 'g')
plot(ursf3)

rs3=zeros(len(lags3))
for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rs3[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf3[k]=1./3*(yrsf3[k-1]+yrsf3[k]+yrsf3[k+1])

rsf31=zeros(len(lags3))

ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)
```

```
for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf31[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf3[k]=1./3*(yrsf3[k-1]+yrsf3[k]+yrsf3[k+1])
rsf32=zeros(len(lags3))
ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)

for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf32[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf3[k]=1./3*(yrsf3[k-1]+yrsf3[k]+yrsf3[k+1])
rsf33=zeros(len(lags3))

ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)

for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf33[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf3[k]=1./3*(yrsf3[k-1]+yrsf3[k]+yrsf3[k+1])
rsf34=zeros(len(lags3))

ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)

for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf34[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)

for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf34[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf3[k]=1./3*(yrsf3[k-1]+yrsf3[k]+yrsf3[k+1])
rsf35=zeros(len(lags3))
```

```
ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)

for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf35[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
####---------Downsampling with M=4---------
lags4=arange(0,70)
yrsf4=yr[::4].copy()
ursf4=u[::4].copy()
N=len(yrsf4)
figure()
plot(yrsf4, 'g')
plot(ursf4)

rs4=zeros(len(lags4))
for lag in lags4:
    yshift=yrsf4[lag:N]
    ushift=ursf4[0:N-lag]
    rs4[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
for k in range(1, N-1):
    yrsf4[k]=1./3*(yrsf4[k-1]+yrsf4[k]+yrsf4[k+1])

rsf41=zeros(len(lags4))
ursf4=ursf4[1:N-1]
yrsf4=yrsf4[1:N-1]
N=len(yrsf4)

for lag in lags4:
    yshift=yrsf4[lag:N]
    ushift=ursf4[0:N-lag]
    rsf41[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf4[k]=1./3*(yrsf4[k-1]+yrsf4[k]+yrsf4[k+1])
rsf42=zeros(len(lags4))
ursf4=ursf4[1:N-1]
yrsf4=yrsf4[1:N-1]
N=len(yrsf4)

for lag in lags4:
    yshift=yrsf4[lag:N]
    ushift=ursf4[0:N-lag]
    rsf42[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf4[k]=1./3*(yrsf4[k-1]+yrsf4[k]+yrsf4[k+1])
rsf43=zeros(len(lags4))
ursf4=ursf4[1:N-1]
yrsf4=yrsf4[1:N-1]
N=len(yrsf4)
for lag in lags4:
    yshift=yrsf4[lag:N]
    ushift=ursf4[0:N-lag]
    rsf43[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
```

```
for k in range(1, N-1):
    yrsf4[k]=1./3*(yrsf4[k-1]+yrsf4[k]+yrsf4[k+1])
rsf44=zeros(len(lags4))
ursf4=ursf4[1:N-1]
yrsf4=yrsf4[1:N-1]
N=len(yrsf4)
for lag in lags4:
    yshift=yrsf4[lag:N]
    ushift=ursf4[0:N-lag]
    rsf44[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf4[k]=1./3*(yrsf4[k-1]+yrsf4[k]+yrsf4[k+1])
rsf45=zeros(len(lags4))
ursf4=ursf4[1:N-1]
yrsf4=yrsf4[1:N-1]
N=len(yrsf4)
for lag in lags4:
    yshift=yrsf4[lag:N]
    ushift=ursf4[0:N-lag]
    rsf45[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

#####--------------Downsampling with M=5---------
yrsf5=yr[::5].copy()
ursf5=u[::5].copy()
N=len(yrsf5)
figure()
plot(yrsf5, 'g')
plot(ursf5)

rs5=zeros(len(lags5))
for lag in lags5:
    yshift=yrsf5[lag:N]
    ushift=ursf5[0:N-lag]
    rs5[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
for k in range(1, N-1):
    yrsf5[k]=1./3*(yrsf5[k-1]+yrsf5[k]+yrsf5[k+1])

rsf51=zeros(len(lags5))
ursf5=ursf5[1:N-1]
yrsf5=yrsf5[1:N-1]
N=len(yrsf5)
for lag in lags5:
    yshift=yrsf5[lag:N]
    ushift=ursf5[0:N-lag]
    rsf51[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf5[k]=1./3*(yrsf5[k-1]+yrsf5[k]+yrsf5[k+1])
rsf52=zeros(len(lags5))
ursf5=ursf5[1:N-1]
yrsf5=yrsf5[1:N-1]
N=len(yrsf5)
for lag in lags5:
    yshift=yrsf5[lag:N]
    ushift=ursf5[0:N-lag]
    rsf52[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
```

```
for k in range(1, N-1):
    yrsf5[k]=1./3*(yrsf5[k-1]+yrsf5[k]+yrsf5[k+1])
rsf53=zeros(len(lags5))
ursf5=ursf5[1:N-1]
yrsf5=yrsf5[1:N-1]
N=len(yrsf5)
for lag in lags5:
    yshift=yrsf5[lag:N]
    ushift=ursf5[0:N-lag]
    rsf53[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
for k in range(1, N-1):
    yrsf5[k]=1./3*(yrsf5[k-1]+yrsf5[k]+yrsf5[k+1])
rsf54=zeros(len(lags5))
ursf5=ursf5[1:N-1]
yrsf5=yrsf5[1:N-1]
N=len(yrsf5)
for lag in lags5:
    yshift=yrsf5[lag:N]
    ushift=ursf5[0:N-lag]
    rsf54[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf5[k]=1./3*(yrsf5[k-1]+yrsf5[k]+yrsf5[k+1])
rsf55=zeros(len(lags5))
ursf5=ursf5[1:N-1]
yrsf5=yrsf5[1:N-1]
N=len(yrsf5)
for lag in lags5:
    yshift=yrsf5[lag:N]
    ushift=ursf5[0:N-lag]
    rsf55[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

##-----------end of sampling and filtering------
```

Correlation_analysis_real_data_coarse_graining.py

```
data=loadtxt('data_whole_year.txt')

u=data[1000:1400,1]   # hot water
yr=data[1000:1400,0]   # bulding (rooms average) temp
#================================
# ORIGINAL SAMPLIG PERIOD dT:
dT=60 #[min]
#================================

lags=arange(0,24)
N=len(yr)
r=zeros(len(lags))
yrf=zeros(N)
#------filtering  signal-------
for lag in lags:
    yshift=yr[lag:N]
    ushift=u[0:N-lag]
    r[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrf[k]=1./3*(yr[k-1]+yr[k]+yr[k+1])
```

```
rcg1=zeros(len(lags))
uf=u[1:N-1]
yrf=yrf[1:N-1]
N=len(yrf)
for lag in lags:
    yshift=yrf[lag:N]
    ushift=uf[0:N-lag]
    rcg1[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
rcg2=zeros(len(lags))
uf=u[1:N-1]
yrf=yrf[1:N-1]
N=len(uf)
for lag in lags:
    yshift=yrf[lag:N]
    ushift=uf[0:N-lag]
    rcg2[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
rcg3=zeros(len(lags))
uf=u[1:N-1]
yrf=yrf[1:N-1]
N=len(uf)

for lag in lags:
    yshift=yrf[lag:N]
    ushift=uf[0:N-lag]
    rcg3[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
rcg4=zeros(len(lags))
uf=u[1:N-1]
yrf=yrf[1:N-1]
N=len(uf)
for lag in lags:
    yshift=yrf[lag:N]
    ushift=uf[0:N-lag]
    rcg4[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
for k in range(1, N-1):
    yrf[k]=1./3*(yrf[k-1]+yrf[k]+yrf[k+1])
rcg5=zeros(len(lags))
uf=u[1:N-1]
yrf=yrf[1:N-1]
N=len(uf)
for lag in lags:
    yshift=yrf[lag:N]
    ushift=uf[0:N-lag]
    rcg5[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
```

```
##---------sampling and filtering-----------
###-------------------------------
lags2=arange(0,15)
yrsf=yr[::2].copy()
ursf=u[::2].copy()
N=len(yrsf)
figure()
plot(yrsf,'g')
plot(ursf)

rsf=zeros(len(lags2))
for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf[k]=1./3*(yrsf[k-1]+yrsf[k]+yrsf[k+1])

rsf1=zeros(len(lags2))
ursf=ursf[1:N-1]
yrsf=yrsf[1:N-1]
N=len(yrsf)
for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf1[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
for k in range(1, N-1):
    yrsf[k]=1./3*(yrsf[k-1]+yrsf[k]+yrsf[k+1])
rsf2=zeros(len(lags2))
ursf=ursf[1:N-1]
yrsf=yrsf[1:N-1]
N=len(yrsf)
for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf2[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf[k]=1./3*(yrsf[k-1]+yrsf[k]+yrsf[k+1])
rsf3=zeros(len(lags2))
ursf=ursf[1:N-1]
yrsf=yrsf[1:N-1]
N=len(yrsf)
for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf3[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf[k]=1./3*(yrsf[k-1]+yrsf[k]+yrsf[k+1])
rsf4=zeros(len(lags2))
ursf=ursf[1:N-1]
yrsf=yrsf[1:N-1]
N=len(yrsf)
```

```
for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf4[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf[k]=1./3*(yrsf[k-1]+yrsf[k]+yrsf[k+1])
rsf5=zeros(len(lags2))
ursf=ursf[1:N-1]
yrsf=yrsf[1:N-1]
N=len(yrsf)
for lag in lags2:
    yshift=yrsf[lag:N]
    ushift=ursf[0:N-lag]
    rsf5[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
###-------------------------------------------------------------
lags3=arange(0,15)
yrsf3=yr[::3].copy()
ursf3=u[::3].copy()
N=len(yrsf3)
figure()
plot(yrsf3, 'g')
plot(ursf3)

rs3=zeros(len(lags3))
for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rs3[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf3[k]=1./3*(yrsf3[k-1]+yrsf3[k]+yrsf3[k+1])

rsf31=zeros(len(lags3))

ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)
for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf31[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf3[k]=1./3*(yrsf3[k-1]+yrsf3[k]+yrsf3[k+1])
rsf32=zeros(len(lags3))

ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)

for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf32[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
```

```
for k in range(1, N-1):
    yrsf3[k]=1./3*(yrsf3[k-1]+yrsf3[k]+yrsf3[k+1])
rsf33=zeros(len(lags3))

ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)

for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf33[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf3[k]=1./3*(yrsf3[k-1]+yrsf3[k]+yrsf3[k+1])
rsf34=zeros(len(lags3))

ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)

for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf34[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)

for k in range(1, N-1):
    yrsf3[k]=1./3*(yrsf3[k-1]+yrsf3[k]+yrsf3[k+1])
rsf35=zeros(len(lags3))

ursf3=ursf3[1:N-1]
yrsf3=yrsf3[1:N-1]
N=len(yrsf3)

for lag in lags3:
    yshift=yrsf3[lag:N]
    ushift=ursf3[0:N-lag]
    rsf35[lag]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
##----------end of sampling and filtering------
```

MSFNA_also_negative_lags_artdata.py

```
u=loadtxt('u.txt')[500:1500]
y=loadtxt('y.txt')[500:1500]

dt=6 # minutes

N=len(y)

ny=5
nu=5

lags=arange(-120,121,10)  # samples f(X(k))=y(k+lag)
CCountMFNN=zeros((len(lags)))

dy=array((0.1,1,10))
dX=array((1000,100,10))
```

```
MFNN=zeros((len(dX),len(dy),len(lags)))    #MFNN[dX,dy]=4x4
Nofeach=N-max(abs(lags))-ny-nu

for ilag in range(len(lags)):
    print "lag = ", lags[ilag]*dt/60.," [hours]"  # f(X)=y(k+lags[ilag])
    X=zeros((N,ny+nu))
    if lags[ilag]<0:
        yout=y[max(ny,nu)-1:N+lags[ilag]]
        krange=range(max(ny,nu)-1-lags[ilag],N)
    else:
        yout=y[max(ny,nu)+lags[ilag]-1:N]
        krange=range(max(ny,nu)-1,N-lags[ilag])

    for k in krange:
        X[k,0:ny]=y[range(k,k-ny,-1)]
        X[k,ny:]=u[range(k,k-nu,-1)]
    X=X[0:Nofeach]
    yout=yout[0:Nofeach]

    for idX in range(len(dX)):
        print "dX=", dX[idX]
        for jdy in range(len(dy)):
            print "dy=",dy[jdy]
            FNN=0
             for i in range(0,len(yout)):
                 for j in range(i+1,len(yout)):
                     dXij=sqrt(sum((X[i,:]-X[j,:])**2))
                     dyij=sqrt(sum((yout[i]-yout[j])**2))
                     if dXij<dX[idX] and dyij>dy[jdy]:
                         FNN=FNN+1
             MFNN[idX,jdy,ilag]=FNN
    CCountMFNN[ilag]=sum(MFNN[:,:,ilag])
    print "CCountMFNN[lag]=", CCountMFNN[ilag]

r=zeros(len(lags))
pom=0
for lag in lags:
    if lag<0:
        yshift=y[0:N+lag]
        ushift=u[-lag:N]
    else:
        yshift=y[lag:N]
        ushift=u[0:N-lag]
    r[pom]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
    pom=pom+1

t=arange(len(y))*dt/60.
```

MSFNA_also_negative_lags_realdata.py

```
u=data[100:1500,1]   # hot water
y=data[100:1500,0]   # bulding (rooms average) temp

dt=60 # minutes

N=len(y)

ny=10
nu=10

lags=arange(-15,16)  # samples f(X(k))=y(k+lag)

CCountMFNN=zeros((len(lags)))

dy=array((0.1,1,10))
dX=array((1000,100,10))

MFNN=zeros((len(dX),len(dy),len(lags)))   #MFNN[dX,dy]=4x4

Nofeach=N-max(abs(lags))-ny-nu


for ilag in range(len(lags)):
    print "lag = ", lags[ilag]*dt/60.," [hours]"  # f(X)=y(k+lags[ilag])
    X=zeros((N,ny+nu))
    if lags[ilag]<0:
        yout=y[max(ny,nu)-1:N+lags[ilag]]
        krange=range(max(ny,nu)-1-lags[ilag],N)
    else:
        yout=y[max(ny,nu)+lags[ilag]-1:N]
        krange=range(max(ny,nu)-1,N-lags[ilag])

    for k in krange:
        X[k,0:ny]=y[range(k,k-ny,-1)]
        X[k,ny:]=u[range(k,k-nu,-1)]
    X=X[0:Nofeach]
    yout=yout[0:Nofeach]

    for idX in range(len(dX)):
        print "dX=", dX[idX]
       for jdy in range(len(dy)):
            print "dy=",dy[jdy]
            FNN=0
            for i in range(0,len(yout)):
                for j in range(i+1,len(yout)):
                    dXij=sqrt(sum((X[i,:]-X[j,:])**2))
                    dyij=sqrt(sum((yout[i]-yout[j])**2))
                    if dXij<dX[idX] and dyij>dy[jdy]:
                        FNN=FNN+1
            MFNN[idX,jdy,ilag]=FNN
   CCountMFNN[ilag]=sum(MFNN[:,:,ilag])
   print "CCountMFNN[lag]=", CCountMFNN[ilag]
```

```
r=zeros(len(lags))
pom=0
for lag in lags:
    if lag<0:
        yshift=y[0:N+lag]
        ushift=u[-lag:N]
    else:
        yshift=y[lag:N]
        ushift=u[0:N-lag]
    r[pom]=1./len(ushift)*(sum((ushift-mean(ushift))*(yshift-mean(yshift))))/std(ushift)/std(yshift)
    pom=pom+1

t=arange(len(y))*dt/60.
```