AUTOREGRESSIVE MODELS AND ARTIFICIAL NEURAL NETWORKS IN TIME SERIES PREDICTION

A. Pavelka and A. Procházka

Institute of Chemical Technology, Department of Computing and Control Engineering

Abstract

The paper is devoted to the presentation possibilities in the MATLAB environment using namely the MATLAB Web Server. For statistical data analysis both MATLAB implemented functions and own algorithms have been used. The main goal of the paper is in analysis of methods of signal prediction by classical methods and using adaptive nonlinear algorithms. Both approaches have been tested by autoregressive models and by feed forward and recurrent artificial neural network model. The autoregressive model is based on SVD and QR methods. The determination of model quality has been verified by information tests including Akaike information criterion and mean squared error.

1 Introduction

Signal prediction is very important tool of information engineering with many practical applications. Its theoretical base always comes out from analysis of historical data. Various methods of signal prediction use autoregressive models, different types of neural networks and tools of system identification and optimization. The following paper presents their possible use in prediction of gas consumption. Matlab Web server has been used for the final presentation of results.

2 Structures for Signal Prediction

2.1 Autoregressive Models

The autoregressive model (AR) [8, 9] estimates the output signal $y(n)$ from the linear combination of n_a previous values of this signal and a random element $e(n)$ using relation

$$y(n) = -a_1 y(n-1) - \ \ldots \ - a_{n_a} y(n-n_a) + e(n) \tag{1}$$

AR model is a special case of a stochastic process which is known as an ARMA model

$$y(n) + a_1 y(n-1) + \ \ldots \ + a_{n_a} y(n-n_a) = e(n) + c_1 e(n-1) + \ \ldots \ + c_{n_c} e(n-n_c) \tag{2}$$

Autocorrelation (3) is a special case of cross-correlation. Algorithm of autocorrelation is based on the mutual multiplication of series $\{x(n)\}$ and $\{x(n+k)\}$.

$$R_{xx}(k) = \sum_n x(n)x(n+k), \qquad \text{for } k = 0, \pm 1, \pm 2, \ldots, \pm K \tag{3}$$

After mutual multiplication we receive series of $(2K+1)$ elements. This autocorrelation function has the highest value for $k = 0$; series $\{x(n)\}$ is self-multiplied. The principle of autocorrelation has been used for selection of the maximum order of AR model.

Singular value decomposition (SVD) is an optimal orthogonal decomposition which has a wide field of applications in rank determination and inversion of matrices, as well as in the modelling, prediction, filtering and information compression of data sequences [5]. Given any $m \times n$ real matrix $\mathbf{A}$, there exists an $m \times m$ real orthogonal matrix $\mathbf{U}$, an $n \times n$ real orthogonal matrix $\mathbf{V}$ and an $m \times n$ diagonal matrix $\mathbf{S}$, such that

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \qquad \mathbf{S} = \mathbf{U}^T\mathbf{A}\mathbf{V}. \tag{4}$$

The decomposition (4) is called the singular values decomposition (SVD). The orthogonal, or QR, factorization expresses any rectangular matrix as the product of an orthogonal or unitary matrix and an upper triangular matrix. The QR decomposition of an $m \times n$ matrix $\mathbf{A}$ with rank p is given by

$$\mathbf{A} = \mathbf{QR} \tag{5}$$

where $\mathbf{Q}$ is an $m \times p$ orthogonal matrix and $\mathbf{R}$ is an $p \times n$ upper triangular matrix. When $m = n$, $\mathbf{Q}$ and $\mathbf{R}$ are square matrices, and $\mathbf{Q}$ is an orthogonal matrix. QRcp (that is, QR with column pivoting) factorization is used to pivote the columns of a matrix in order of maximum Euclidian norm in successive orthogonal directions, while QR factorization is performed on the matrix [5]. The symbol cp can be named as column permutation as well. In both cases result is the same.

$$\mathbf{AP} = \mathbf{QR} \tag{6}$$

Together with matrices $\mathbf{Q}$ and $\mathbf{R}$ we receive permutation matrix $\mathbf{P}$ with size $n \times n$. This matrix consists of n ones only.

2.2 Linear Neural Network

The main characteristic feature of the linear neural network (Fig. 1) is its transfer function, which is unlimited linear function with unknown slope. In most cases, we can calculate a linear network directly, such that its error is a minimum for the given input vectors and targets vectors. In other cases, numerical problems prohibit direct calculations. Fortunately, we can always train the network to have the minimum error by using the Least Mean Squares (Widrow-Hoff) algorithm [2]. And thereby this neural network is by own concept very close to the autoregressive models. For the prediction of time series it is suitable to use network architecture with delays. Those delay are already in the ground of time series – in its time dependence.

2.3 Feed-Forward Backpropagation Neural Network

Both a feed-forward backpropagation (Fig. 2) and a linear neural network have no special architecture in comparison with recurrent neural network. For our purpose we used two layer network. The first transfer function was hyperbolic tangent sigmoid function and the second transfer function was unlimited linear function with unknown slope. The Levenberg-Marquardt backpropagation method has been used as a basic learning algorithm for this network type. [1, 3]. This algorithm is based on calculation of the Jacobian matrix:

$$J(\underline{a}) = \begin{bmatrix} \frac{\partial e_1(\underline{a})}{\partial a_1} & \frac{\partial e_1(\underline{a})}{\partial a_2} & \cdots & \frac{\partial e_1(\underline{a})}{\partial a_n} \\ \frac{\partial e_2(\underline{a})}{\partial a_1} & \frac{\partial e_2(\underline{a})}{\partial a_2} & \cdots & \frac{\partial e_2(\underline{a})}{\partial a_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N(\underline{a})}{\partial a_1} & \frac{\partial e_N(\underline{a})}{\partial a_2} & \cdots & \frac{\partial e_N(\underline{a})}{\partial a_n} \end{bmatrix} \tag{7}$$

and its application in the Levenberg-Marquardt modification of backpropagation

$$\Delta \underline{a} = (J^T(\underline{a})J(\underline{a}) + \mu I)^{-1} J^T(\underline{a})\underline{e}(\underline{a}) \tag{8}$$

where $\underline{a} = [w1(1,1)\ w1(1,2)\ \cdots\ w1(S1,R)\ b1(1)\ \cdots\ b1(S1)\ w2(1,1)\ \cdots\ bM(SM)]^T$ is parameter vector, I is identity matrix, μ is learning parameter, $\underline{e}(\underline{a})$ is error vector and M stands for the number of network layers.

2.4 Recurrent Neural Network

A recurrent neural network (Fig. 3) is one in which the outputs from the output layer are fed back to the set of input units. Neural networks of this kind are able to store information about time, and therefore they are particularly suitable for forecasting applications. However, while recurrent neural networks have many desirable features, there are practical problems in

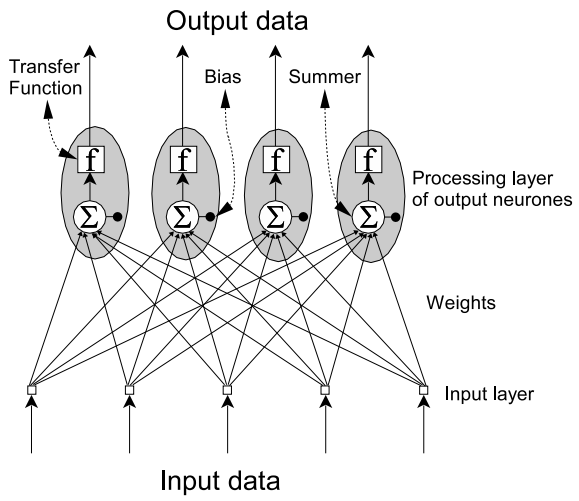

Figure 1: Architecture of linear network and structure of the neuron

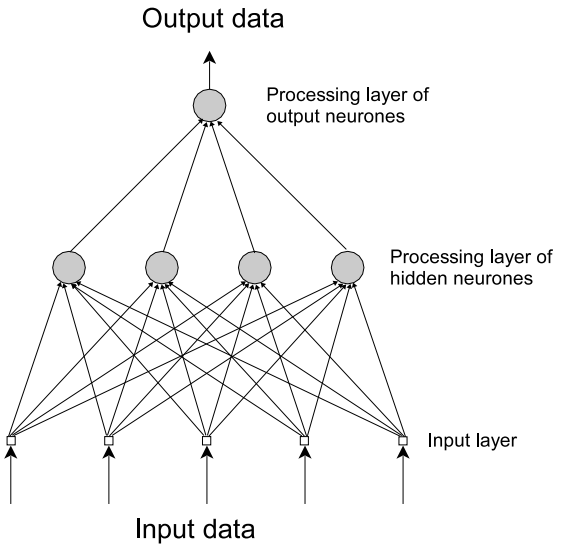

Figure 2: Architecture of two layer feed forward network

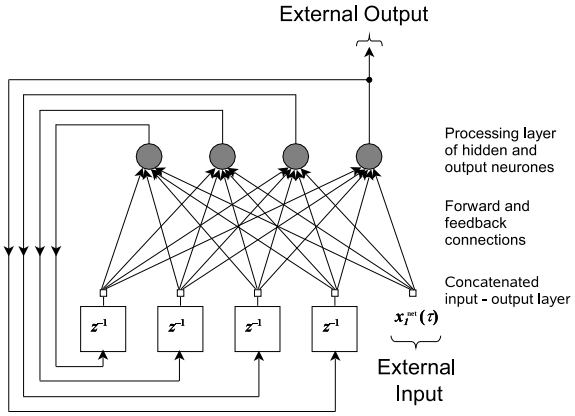

Figure 3: Architecture of recurrent network with one external input and one output [4]

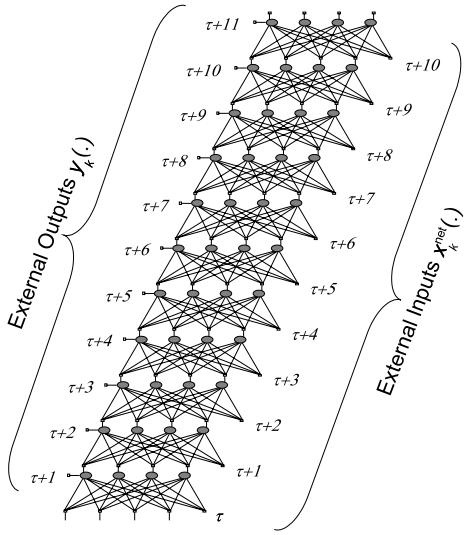

Figure 4: In time unfolded recurrent network with 11 time layers

developing effective training algorithms for them [7]. For prediction we used a one layer model with variable number of external inputs and with a variable number of hidden neurones. Hidden neurones in architecture of the recurrent network are neurones without external input (Fig. 3). The algorithm used for computing the error gradient is a so-called *epochwise backpropagation through time* [12, 13], which can be derived by unfolding the temporal operation of a network into a multi layer feed forward network that grows by one layer each time step (Fig. 4). This algorithm is organized as follows. With t_0 denoting the start time of the epoch and t_1 denoting its end time, the objective is to compute the gradient of $E^{\text{total}}(t_0; t_1)$. This is done by first letting the network run through the interval $[t_0; t_1]$ and saving the entire history of inputs to the network, network state, and target vectors over this interval. Then a single backward pass over this history buffer is performed to compute the set of values of local gradients $\delta_k(\tau) = -\partial E^{\text{total}}(t_0, t_1)/\partial s_{kl}$, for all $k \in U$ and $\tau \in (t_0; t_1]$, by means of the equations

$$\delta_k(\tau) \;=\; \begin{cases} f'_k(s_k(\tau))e_k(\tau) & \text{if } \tau = t_1 \\ f'_k(s_k(\tau))\Big[e_k(\tau) + \sum_{l\in U} w_{lk}\delta_l(\tau+1)\Big] & \text{if } t_0 < \tau < t_1. \end{cases} \tag{9}$$

The symbol U denotes the set of output neurones and $e_k(\tau)$, $k \in U$, are the corresponding errors. Once the backpropagation computation has been performed back to time $t_0 + 1$, the weight changes may be made along the negative gradient of overall error by means of the equations

$$\triangle w_{kl}(\tau) = -\alpha \frac{\partial E^{\mathrm{total}}(t_0, t_1)}{\partial w_{kl}} = \alpha \sum_{\tau=t_0+1}^{t_1} \delta_k(\tau) x_l(\tau - 1) \tag{10}$$

where α is a positive learning rate parameter. Then a new weight matrix $w_{kl}(\tau + 1)$ is counted

$$w_{kl}(\tau + 1) = w_{kl}(\tau) + \triangle w_{kl}(\tau). \tag{11}$$

3 Details of Used Prediction Methods

3.1 Data Specification and Spectral Analysis

The data analysis has been applied to two data sets of gas consumption in the Czech Republic. The first one stands for the data sequence measured with the sampling period of two hours (1. Jan 1997 – 31. Mar 2000) and the second one represents the average gas consumption per one day (1. Jan 1994 – 31. Dec 2001). For the same time period (1. Jan 1994 – 31. Dec 2001) the average daily temperature is available (Fig. 5). The study of the given data resulted in the fact that some measurements were not performed or they are missing. For the future work with time series it was important to fill in the missing measurements. The simplest way has been used: missing values were replaced by the linear interpolation of the closest values.

The initial data analysis has been applied to normalized data modified by difference of logarithms resulting in the sequence $x_i = \log(x_{i+1}) - \log(x_i)$. Spectral analysis has been performed by the algorithm of FFT, which is built in MATLAB environment as a `fft` function. Spectral density graph (Fig. 6 — graphs does not show the whole range of frequencies) calculated from the data modified by logarithmic difference clearly shows meaningful periodicity of 2 hours, one day, 12 hours and 8 hours. In the low frequency area we find periods 3.5 days and 7 days and of course period on one year. Results of spectral analysis of periods occurred in evolution of average daily temperature in the Czech Republic were not so clear (Fig. 6c). Results of those analysis have been used in selecting different empiric constants.

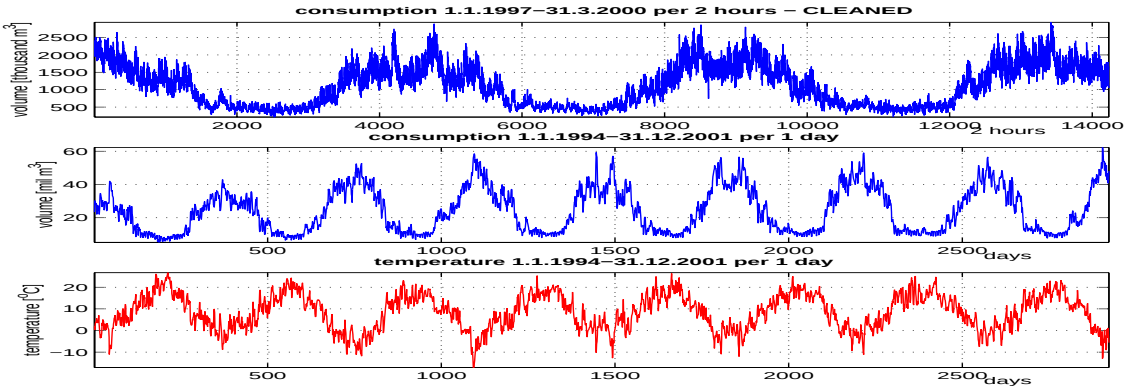

Figure 5: Three original time series before trend removing and standardization

3.2 Data Pre-processing for all Models

For all following calculations only the data with periodicity of one day has been used. All data have been normalized by the trend removing, subtraction of mean value, dividing by the standard deviation and finally re-scaling into the interval $-1 \leq x \leq 1$. After calculations

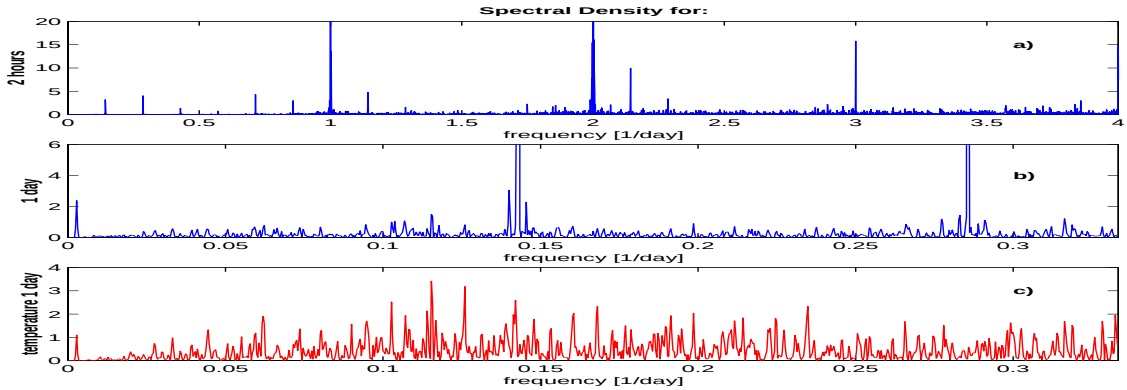

Figure 6: Spectral density of gas consumption and temperature using difference of log

(prediction results), for the higher information value, all data sets have been re-calculate back into the scale of real data of gas consumption in the Czech Republic. Data sets modified by the difference of logarithm have not been used for the prediction.

3.3 Common Methods

As the *classical* prediction model we understand model, for which the whole data set is divided in two parts. The first part is used to estimation of parameters and the second part is used for the model validation. The *adaptive* prediction model takes necessary amount of data, calculates parameters and gives the result, then the data set needful for calculation of parameters is shifted (usually 1 sample ahead), parameters are recalculated and we receive new next result, and so on. The process of model creation has been based upon various methods [6, 10, 11].

Results of calculations are presented in the table presented in Fig. 7 available on the Web page http://phobos.vscht.cz/pavelkaa/. This table presents selected statistical characteristics (*mean*, *standard deviation*, *minimal* and *maximal* values) describing properties of errors calculated as diference between real data of gas consumption and the output from models. Information criteria SSE and MSE describe the whole model. The number of values for which the distance between real and predicted value is less or higher than 5% of nominal value of real data is in percentage mentioned in columns called as *in 5%* and *out 5%*. Our desire is to have maximal value in column *in 5%*. In other words, the column *in 5%* give us the information how many predicted values in percentage have error less than 5% of its real nominal value.

3.4 Autoregressive Model

The number of parameters for the full AR model can be received from the autocorrelation analysis of the given data. Searching algorithm has been applied on the standardized data.At first the autocorrelation function has been calculated and then, to remove pertinent periodicity, the float mean has been calculated. The number of parameters for the full AR model has been set equal to index, where autocorrelation function modified by float mean, had its 10% descent. According to those conditions the full AR model should have n parameters applied to *inputs with gas values*, *inputs with temperature values* and *Day Identifier* (unchecked is 0, checked is 1). The identification of the best subset AR model can be performed through the following three steps:

- Perform SVD of $m \times n$ matrix $\mathbf{A}$ ($\mathbf{A} = \mathbf{USV}^T$). Choose the possible pseudorank of $\mathbf{A}$ from the magnitude of the singular values. For each pseudorank g ($2 \leq g \leq n-1$), perform QRcp factorization on the $g \times n$ matrix $\mathbf{V}^T$ and select the set of relatively independent regressor variables of size g; so subset S_g of regressors corresponding to each pseudorank g is defined.
- For each subset AR model information criterion is computed. The subset S_r corresponding to pseudorank r for which information criterion attains the optimal value (in most cases the minimum value), is selected; the regressors of S_r are the candidates for the desired best subset AR model.

- For the $(2^r - 1)$ models, with all possible combinations of regressors, information criterion is determined. The model producing for example the minimum value for AIC is the best subset AR model.

For the selection of number of parameter we can use one of five possibilities:

1. meaningful difference among singular values
2. descent of MSE in 5% surround of value MSE for the full set model
3. descent of re-calculated MSE in 5% surround of value MSE for the full set model
4. descent of AIC in 5% surround of value AIC for the full set model
5. descent of re-calculated AIC in 5% surround of value AIC for the full set model

Possible number of parameters for the selected models is summarized in the table presented in Fig. 7, where in column *diag S* we find recommended numbers of parameters (values of gas consumption) according to values gained from diagonal matrix **S** from SVD, other columns likewise but with appropriate to information criterion. Item *rc* in the table signed Re-Calculated, this is product of our modification in using SVD and QRcp. After applying of permutation matrix **P** we can use real permutated parameters or we can calculate new ones. After that can have two different models with different accuracy. Usually results from the re-calculated models are better.

3.5 Linear Neural Network

One-layer linear network with arbitrary architecture has been designed by MATLAB's instruction `newlind` from the Neural Network Toolbox. For learning of this network the overdetermined system of equations has been defined and solved by the least-square method (LSM). Among positives of this method belong its calculation time, good accuracy and transparentness.

3.6 Feed-Forward Backpropagation Neural Network

The two-layer network created by MATLAB command `newff` of Neural Network Toolbox is using Levenberg-Marquardt backpropagation [3] as the learning algorithm. The network has an arbitrary architecture, 5 training epochs, hyperbolic tangent sigmoid as the first transfer function and unlimited linear function with unknown slope as the second one. All networks parameters – number of hidden neurones, number of training epochs and transfer functions – have a strong influence to the final prediction results. Their mutual combination resulting in better prediction properties form one of the futures goals.

We have to emphasize that problem of suitable parameters estimation is not in the area of decreasing learning error, it is relatively easy to train the network to minimize this error. The more important problem is in the application of the resulting network for real data not used in the learning stage. Networks seem to be in most cases over-learned and not able to generalize resulting in the increase of error values. To minimize this problem several calculations have been carried out to find suitable parameter regime with such items as number of hidden neurones (from 1 to 50), number of learning epochs (in the range 5-500).

3.7 Recurrent Neural Network

Recurrent neural network which has been used as the last prediction model is relatively highly sophisticated and complicated neural network. The network has been designed with arbitrary architecture, too. As a squashing[1] function the hyperbolic tangent has been used and the epochwise backpropagation through time [12] as the learning algorithm has been used. Every network with specific architecture had 500 learning epochs. To find typical behaviour for every single network architecture one learning cycle three times with 3 different initial weight initialization has been done, so for for one specific network we have 3 final results. From them the best one has bee

[1] transfer

selected with the lowest learning error represented by the SSE criterium. The learning algorithm of recurrent network have specific safeties against finding the global minimum in area of local minimum in the re-initialization of the continuously decreasing learning error. Considering the circumstances that the algorithm of recurrent network with epochwise backpropagation through time algorithm is not included in Neural Network Toolbox of the MATLAB environment there was a demand to programme it by ourself.

Calculations of recurrent network takes relatively long calculation time in order of minutes. Results of this network were often better comparing with the simple network architecture.

4 Matlab Web Server

All final algorithms and codec have been rewritten and implemented into Matlab Web Server running at our department. We widely recommend to visit http://phobos.vscht.cz/pavelkaa where simple but clear web pages are settled. Simple and easy design have been created is aspect to relatively long calculation times. Everyone can test own imagination of networks architectures, own demands on accuracy and precision of prediction. You find here helpful off-line and on-line documentation, smart error checking and much more.

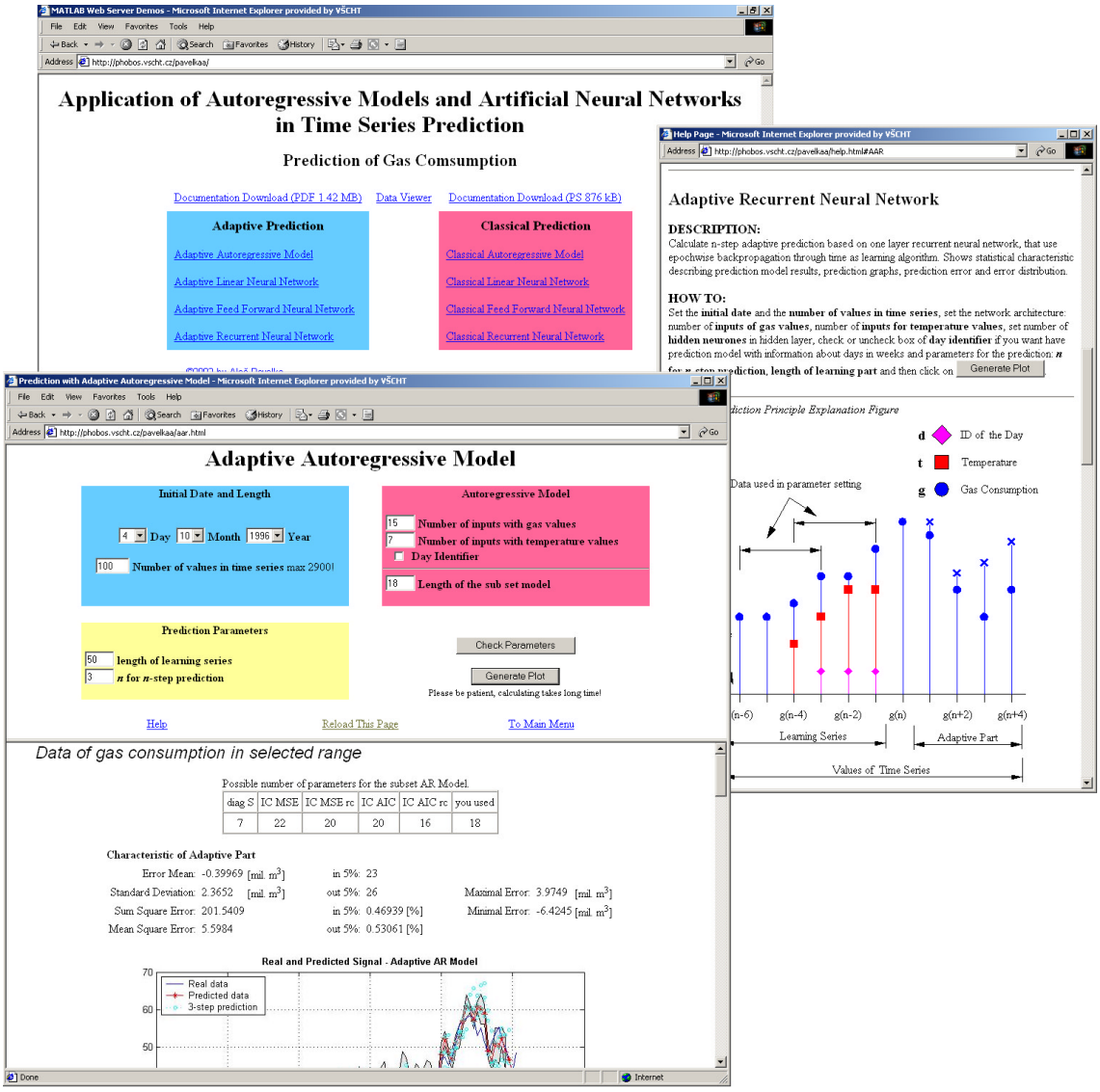

Figure 7: Interactive Web (model) interface

5 Conclusion

All presented prediction methods (linear, feed-forward and recurrent neural networks) have been applied to real data sets of gas consumption in the Czech Republic during winter periods. According to all previous tests it seems that the best prediction method is represented by the linear neural network. But we have to remind, that this research has been used just for modelling prediction possibilities of mentioned prediction methods and their simple comparison. For the true specification and decision which method is better than the others it will be necessary to make more prediction tests.

There are many ways how to improve the prediction accuracy and precision. For example by adding another data into the models like a data of wind speed, gas prices and so on. Or we can use possibilities of more prediction methods and combine their results.

There are still many unanswered questions that should be studied. It is assumed that the future research will be devoted to

- prediction testing using different kinds of preprocessing
- selection of different data sets and applying them in own algorithms
- multi-step prediction by improving (or modification) of a one-step prediction model

References

[1] Anonymous. *The MathWorks-Online Documentation (Help Desk).* The MathWorks, Inc., Natick, MA, 12.1 edition, 2001. http://www.mathworks.com.

[2] Howard Demuth and Mark Beale. *Neural Network Toolbox.* The MathWorks, Inc., Natick, MA, 7 edition, March 2001. http://www.mathworks.com.

[3] Martin T. Hagan and Mohammad B. Menhaj. Training feedforward networks with the marquadt algorithm. *IEEE Transactions On Neural Networks*, 5(6):989–993, 1994.

[4] S. Haykin. *Neural Networks.* Prentice Hall, Inc., New Jersey, 1994.

[5] P. P. Kanjilal. *Adaptive Prediction and Predictive Control.* Peter Peregrinus Ltd., London, 1995.

[6] A. Khotanzad, H. Elragal, and T. L. Lu. Combination of neural-network forecasters for prediction of natural gas consumption. *IEEE Transactions On Neural Networks*, 11(2):464–473, March 2000.

[7] N. Kotzé. Chapter neural networks. WWW, 1999. http://www.ee.up.ac.za/ee/ telecoms/telecentre/ nicchapternn.html.

[8] Lennart Ljung. *System Identification: Theory for the User.* Prentice-Hall, Inc., New Jersey, 1987.

[9] Lennart Ljung. *System Identification Toolbox User's Guide.* The MathWorks, Inc., Natick, MA, 5 edition, April 2001.

[10] A. Procházka. Neural networks and seasonal time-series prediction. *In Fifth International Conference on Artificial Neural Networks Cambridge, England*, 1997.

[11] Martin Šorek and Aleš Procházka. Neural networks in signal prediction. *Signal Analysis and Prediction*, I:228–231, June 1997. The First European Conference on Signal Analysis and Prediction.

[12] R. J. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2:490–501, 1990. http://www.ccs.neu.edu/ home/rjw/pubs.html.

[13] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989. http://www.ccs.neu.edu /home/rjw/pubs.html.

Prof. Aleš Procházka, Ing. Aleš Pavelka
Institute of Chemical Technology, Prague
Department of Computing and Control Engineering
Technická 1905, 166 28 Prague 6 Phone.: 00420-2-2435 4198, Fax: 00420-2-2435 5053
E-mail: A.Prochazka@ieee.org, ales.pavelka@volny.cz