computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific Journal of Computing

# IMPROVED MODEL ORDER ESTIMATION FOR NONLINEAR DYNAMIC SYSTEMS

**László Sragner [1], Gábor Horváth [2]**

Department of Measurement and Information Systems, Technical University of Budapest
H-1117 Budapest, Magyar tudósok körútja 2.
1) sragner@mit.bme.hu, 2) horvath@mit.bme.hu

**Abstract:** *In system modeling the choice of proper model structure is an essential task. Model structure is defined if both the model class and the size of the model within this class are determined. In dynamic system modeling model size is mainly determined by model order. The paper deals with the question of model order estimation when neural networks are used for modeling nonlinear dynamic systems. One of the possible ways of estimating the order of a neural model is the application of Lipschitz quotient. Although it is easy to use this method, its main drawback is the high sensitivity to noisy data. The paper proposes a new way to reduce the effect of noise. The idea of the proposed method is to combine the original Lipschitz method and the Errors In Variables (EIV) approach. The paper presents the details of the proposed combined method and gives the results of an extensive experimental study.*

**Keywords:** - *Model order, Errors-in-Variables, Lipschitz-method*

## 1. INTRODUCTION

For building a system model many different approaches can be used depending on the main features of the system, the purpose of the model and the available information about the system to be modeled. If model building is based mainly on input-output data, black box models can be constructed. As building black box models may be much simpler than physical ones, they are used not only when the lack of physical insight does not let us build physical models, but also in such cases when we have enough physical knowledge, but it is too complex, there are mathematical difficulties, or when the cost of building physical models is too high, etc.

In black box modeling a general model structure must be selected, which is flexible enough to build models for a wide range of different systems.

In black box modeling neural networks play important role. As there are neural networks, like multi- layer perceptrons (MLP) and radial basis function (RBF) networks that are universal approximators ([1], [2]), neural networks can be considered as general modeling devices. Although the basic neural architectures can be applied only for modeling static systems, they can be extended easily such that they are able to model dynamic systems as well.

For dynamic modeling local memory (e.g. tapped delay lines) and/or feedback must be applied. Using different extensions different classes of dynamic neural models can be constructed such as NFIR, NARX, NOE, NARMAX, etc. [3].

For selecting the model structure first one has to choose a model class then the order of the model. E.g. if the NFIR model class is selected it uses the past values of the input data, so it applies a tapped delay line in the input data path and we have to determine the length of this tapped delay line. Similarly for a NARX or a NOE model the lengths of the corresponding input and output tapped delay lines must be determined.

As the correct model order is often not known a priori it makes sense to postulate several different model orders. Based on these, some criterion can be computed that indicated which model order to choose.

To determine the model order is not any easy task. Various methods and criterions have been developed for estimating the model order. These criteria like Akaike Information Criterion (AIC) [4], Final Prediction Error (FPE) [5], or Minimum Description Length (MDL) [6], etc. were developed for linear systems, however most of them can be applied for neural networks too. Using these criteria we can only qualify the models after the construction and the validation of the model, so it is possible that we have to train several neural

networks to get the best model order. Recently a different approach was suggested by He and Asada [7], where the estimate of the model order is determined directly from the measurement data. This approach is based on the continuity property of the nonlinear function, which represents the input-output model of a continuous dynamic system. The approach calculates Lipschitz quotients of the system input-output mapping using solely the measured input-output data. Based on the Lipschitz quotients a Lipschitz number can be defined. The Lipschitz number can indicate whether or not the proper order of the model has been found.

The problem with the Lipschitz method is that it does not give a sharp answer in noisy cases. If the measurement data are noisy we can estimate only a range where the proper model order can probably be found. In this case further search within this range is required, so the Lipschitz number can be used only for getting a good starting point for the order estimation. The correct model order can be determined only if several different models with model order in this range are built. In neural modeling this means that we have to train and validate several dynamic neural networks of different complexity.

To get an easier way of obtaining the proper model this paper proposes a new approach: the Lipschitz method can be combined with the training of the networks using the Errors-In-Variables (EIV) cost function [8].

The application of Error-In-Variables cost function during training makes it possible not only to determine the parameters (weights) of the neural model, but to optimise the noisy input data. The EIV approach takes into consideration the statistical features of the noisy training data, so EIV let us obtain a new, less noisy training set. Based on this training set the Lipschitz method can result in a more accurate estimation of the model order. The proposed method needs the calculation of the Lipschitz number at least two times. If the second run of the Lipschitz method gives sharper results than in the first case we know that the order what we selected was good, otherwise we have to select another one. The efficiency of the combined method has been tested by extensive experiments. The results show that the greatest improvement occurs at the correct order. The combined method can be applied for different model classes. Here it will be presented for NARX models only.

The paper is organized as follows. In the second section we will give a brief overview of the different nonlinear dynamic model classes and we will give some arguments about the importance of the NARX model class. In the third and fourth sections we will describe shortly the Lipschitz method and the basic idea of using EIV cost function, respectively. Section V. deals with the proposed combined method. Finally, section VI presents the results of the experimental study.

## 2. BACKGROUND

There are several ways to form dynamic neural networks using static neurons, however in all ways we use storage elements and/or apply feedback. Both approaches can result in different dynamic neural network architectures.

Storage elements can be used in different parts of a static network. For example some storage modules can be associated with each neuron, with the inputs or with any intermediate nodes of a static network. As an example a feed-forward dynamic network can be constructed from a static multi-input – single-output network (e.g. from an MLP or RBF) if a tapped delay line is added to its inputs. This means that the static network is extended by an embedded memory, which stores the past values of the inputs.

A more general form of constructing dynamic neural networks is if first we define a regressor vector, and this vector will be used as input of a static neural network. Thus the output of the neural model is described as a parameterized nonlinear function of this regressor vector [3]:

$$y(t) = f(x(t), \Theta) \tag{1}$$

Here $H$ is the parameter vector and $x(t)$ denotes the regressor vector.

The regressor can be formed from past inputs, past system outputs, past model outputs etc. according to the model structure selected. When only the past inputs are used as the components of the regressor vector an NFIR model is obtained. If both past inputs and past system outputs are used in the regressor the NARX model can be constructed. Here the regressor vector is formed as:

$$x(t) = [d(t-1), d(t-2), \ldots, d(t-m), u(t), u(t-1), \ldots, u(t-l)] \tag{2}$$

and the mapping of the dynamic neutral network is described by

$$y(t) = f(d(t-1), d(t-2), \ldots, d(t-m), u(t), u(t-1), \ldots u(t-l), \Theta) \tag{3}$$

where $y(t)$ is the calculated (model) output, $d(t)$ is the system output (the desired output in the training process) and $u(t)$ is the system input at the *t*-th time step. Similarly, defining the regressor vectors properly we can form NOE, NARMAX or NJB model classes [3].

In this paper we will deal only with the NARX model class. The advantage of this model class is that its inputs (the regressor vector) are directly taken from the system input-output data. It can realize dynamic models; it works as if it were a feedback system, while during training it can be handled as a feedforward network, avoiding any stability problem. Of course the results are valid for NFIR models too, as it can be considered as a special NARX model without feedback part.

For NARX models the model order is defined as the lengths of the tapped delay lines, the values of *m* and *l* in Eq. (2). It means how many lagged inputs and outputs are taken into account in the inputs of the static neural network. If the model order is (*m,l*) and the system to be modeled has *a*-dimensional input and *b*-dimensional output, then the corresponding static neural network will have $mb + l(a+1)$ inputs.

## 3. THE LIPSCHITZ METHOD

If a model is constructed solely from input-output data, both model structure and the model parameters should be determined using these data. The Lipschitz method applies a direct way of obtaining the model order from the available input-output data. The parameters of the model, the weights of the dynamic neural network will be determined from the same data set during training.

The basic idea behind the Lipschitz method is the Lipschitz theorem, which tells that every continuous mapping has bounded gradient, which can be estimated by the maximum of the gradients at the known points. Assuming that the system to be modeled implements a smooth input-output mapping this idea can be used for getting an estimate of model order. The Lipschitz method determines such numbers - the Lipschitz numbers - that measure the smoothness of the mapping. The Lipschitz numbers are based on the following quotient:

$$L_{t_1,t_2}^{(m,l)} = \frac{\left\|d(t_2)-d(t_1)\right\|^2}{\sqrt{\frac{1}{mb}\sum_{p=1}^{m}\left\|d(t_2-p)-d(t_1-p)\right\|^2+\frac{1}{(l+1)a}\sum_{q=0}^{l}\left\|u(t_2-q)-u(t_1-q)\right\|^2}} \quad (4)$$

where $\|.\|$ is the Euclidean norm of a vector. Eq. (4) determines the normalized slope of the mapping between two points of the training set. From the $L_{t_1,t_2}^{(m,l)}$ quotients the Lipschitz number can be calculated as:

$$L^{(m,l)} = \left(\prod_{k=1}^{p}\sqrt{mb+(l+1)a}\,L^{(m,l)}(k)\right)^{\frac{1}{p}} \quad (5)$$

where $L^{(m,l)}(k)$ is the *k*-th largest quotient among all $L_{t_1,t_2}^{(m,l)}$ $t_1 \neq t_2$. This is a geometrical average weighted by the model order and the number of input and output dimensions (the number of the neural network inputs). The number of averaged values is denoted by *p*; this is usually 1-2 percent of the size of the training set.

From the continuity property it follows that if $(m_0,l_0)$ is the optimal order-pair, the $L^{(m,l)}$ Lipschitz numbers at $(m_0+1,l_0)$, $(m_0,l_0+1)$ and $(m_0+1,l_0+1)$ are nearly the same as at $(m_0,l_0)$, but at $(m_0-1,l_0)$, $(m_0,l_0-1)$ and $(m_0-1,l_0-1)$ are much larger than $L^{(m_0,l_0)}$.

To get the Lipschitz estimate of the order of a given model we have to calculate Eq. (5) for every possible presumed model order. The optimal order is where the graph of *L* versus *m* and *l* has a sharp breakpoint.

In noisy cases the values of Eq. (4) may vary around the true value because of the noise. In this case the graph of *L* will not have a sharp breakpoint; there will be a steep slope at small orders and a saturation region for large orders but there is a smooth region between them. The optimal model order can be found somewhere in this smooth region. The consequence of this smooth transition from the steep slope to the saturated part is that we cannot get the correct order directly using the Lipschitz method; however we can restrict the possible presumed model orders to a limited and rather small region. To select the optimal order within this region we have to create different order models. We do this with a special training of the network, when during training we have an additional task: to reduce the noise of the input data. After training a second run of the Lipschitz-method will evaluate the selected model order whether it was good or not. The whole process starts with a model of small order and model complexity is increased step-by-step.

## 4. THE ERRORS-IN-VARIABLES COST FUNCTION

To reduce the effects of input noise, which decreases the order-indicating property of the Lipschitz method, we will use the Errors-In-Variables (EIV) approach [9]. The EIV approach applies a modified cost function during the network training. This makes it possible not only to train the weights of the neural network, but to modify the input data for getting a less noisy, more accurate input data set. EIV can be applied if we have some information about the input and output noise: we

have prior information about the variances of the noise components of the data, or multiple measurements can be taken in all time steps, and from these measurements the noise variances can be estimated. The EIV approach applies for MIMO (Multiple Inputs Multiple Outputs) systems; however, to simplify the notations we discuss it for SISO (Single Inputs Single Outputs) systems only.

We seek a black box model for the system in the form $y(t) = f(x(t), \Theta)$. The EIV cost function is the following:

$$C_{EIV} = \frac{1}{N}\sum_{t=1}^{N}\left(\frac{(y(t) - f(x(t),\Theta))^2}{\sigma_y^2(t)} + \frac{(x(t) - x(t))^2}{\sigma_x^2(t)}\right) \quad (6)$$

where $N$ is the number of the time steps, $\hat{y}(t)$ and $\hat{x}(t)$ are the estimated means of the outputs and the inputs of the measurements respectively in the $t$-th time step and $\hat{\sigma}_x^2(t)$ and $\hat{\sigma}_y^2(t)$ are the estimated variances of the input- and output-noise in the $t$-th time step. With this cost function the significance of the contribution of the training samples are determined according to their noise levels. If the noise is greater the training sample affects the total cost less, because we assume that the knowledge obtained from this training is less accurate. The parameters are modified by the following equations using gradient approach:

$$\Delta\Theta_j = \eta\frac{1}{2N}\sum_{t=1}^{N}\frac{y(t) - f(x(t),\Theta)}{\sigma_y^2(t)}\frac{\partial f(x(t),\Theta)}{\partial\Theta_j} \quad (7)$$

$$\Delta x(t) = \eta\frac{1}{2}\left(\frac{y(t) - f(x(t),\Theta)}{\sigma_y^2(t)}\frac{\partial f(x(t),\Theta)}{\partial x_k(t)} + \frac{x(t) - x(t)}{\sigma_x^2(t)}\right) \quad (8)$$

where $\eta$ is the training factor. Eq. (8) shows that we modify not only the parameters of the neural network, but also the input data of the training set. This means that the number of adjusted parameters, the number of free parameters of the whole training process is increased significantly. The increased number of the free parameters could result in overtraining, as a model structure with too many free parameters is rather prone to overfitting. To avoid overtraining early stopping can be used or a starting neural model can be constructed using the classical training process with LS cost function. In our experiments this latter method was applied.

## 5. THE COMBINATION OF THE METHODS

Lipschitz method is an easy-to-use way for getting an estimate of the model order directly from the training data. Its applicability, however, is limited if the input data are noisy. At the same time the main task of EIV is to correct the input data while the network is trained. So combining EIV and Lipschitz method we can correct the noisy data and using the corrected data the Lipschitz method can give a more accurate estimate of the model order.

To utilize the advantages of both methods we use them in the following way: first we make initial model order estimation with the Lipschitz method where the original (noisy) data set is used. The initial estimation does not give an exact answer, instead only a range of possible orders can be determined from the Lipschitz curve. We expect that if we create a model with an order selected from this range and train the neural network using the EIV approach than the noise will reduce. This can be justified by using the Lipschitz-method again. The test results show that the second run of the Lipschitz method gives better results than in the first case, the Lipschitz curve will be sharper near the true model order. The greatest improvement can be measured if the first order-selection was the optimal one. To get the best features of both techniques, we combine the two methods in the following way:

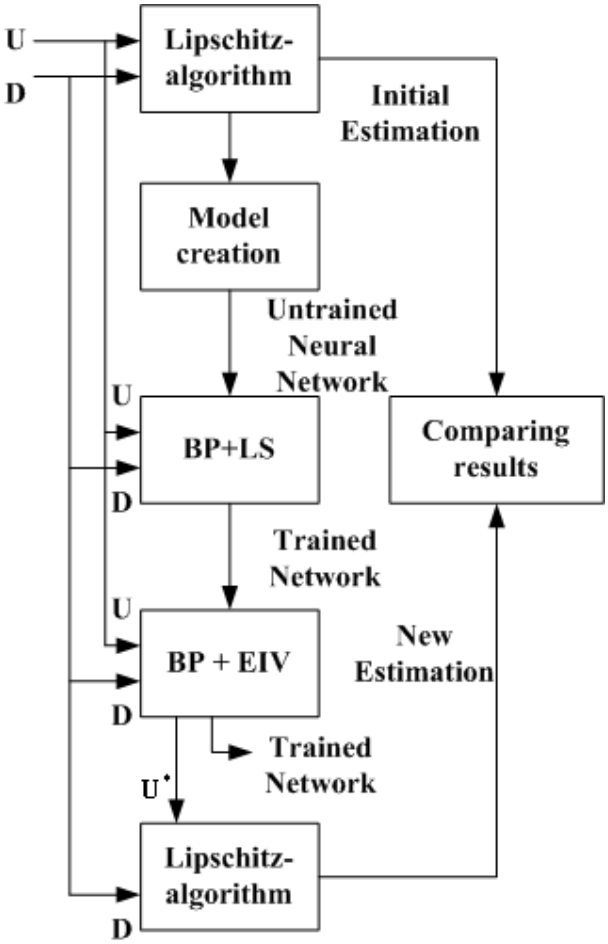


**Fig. 1 – Dataflow diagram of the combined method.**

First we create a model according to the first run of the Lipschitz-method. Next we train it with the

classical supervised learning which is based on LS cost function. The task of this early training is to lower the effect of overtraining by getting a rough estimate of the parameter vector. This will give a good starting point for the EIV approach. The BP (backpropagation) and EIV finalize the parameters of the NN and modify the input sequence. With this modified sequence and the original output sequence we calculate the second Lipschitz estimates, and compare the results with those of the first run. If noise is present, there will be no sharp breakpoint in the Lipschitz-graph, so we have to select the model order from a neighbourhood where the optimal model order is most likely to be, than we can repeat the parameter estimation sequence with another model order selected from this neighbourhood. The optimal model order is the one where the greatest improvement of the Lipschitz estimates can be achieved.

## 6. TEST RESULTS

The combined method was tested on several artificial problems. Here only the results of a SISO (Single Input Single Output) model with model order (0, 3) will be presented. In this case there is no feedback from the output to the input and three previous inputs are used as inputs of a static neural network. The transfer function of the test problem is the following:

$$y(t) = f(u(t), u(t-1), u(t-2), u(t-3)) \tag{9}$$

$$\begin{aligned} &f(x_0, x_1, x_2, x_3) = \\ &th(10x_0 + 4) - th(10x_0 + 3) + th(10x_0 - 4) - th(10x_0 - 3) + \\ &th(10x_1 + 6) - th(10x_1 + 5) + th(10x_1 - 6) - th(10x_1 - 5) + \\ &th(10x_2 + 5) - th(10x_2 + 4) + th(10x_2 - 5) - th(10x_2 - 4) + \\ &2(th(10x_3 + 7) - th(10x_3 + 6) + th(10x_3 - 7) - th(10x_3 - 6)) \end{aligned} \tag{10}$$

1000 random sample points were generated from a uniform distribution in [-1,+1]; the same input-output data set was used in every experiments. The input data were corrupted by additive Gaussian noise with 0 mean and 0.01, 0.05, 0.1 and 0.2 standard deviations. For obtaining estimates of the noise variances that are is necessary for using Eq. (6), we repeat the measurements by 3-, 5- and 10-times.

In the noiseless case the true order could be obtained from the Lipschitz numbers, but in the noisy cases it cannot be decided whether the optimal order was (0, 2), (0, 3) or (0, 4). Some experiments were also done using model orders more different from the true values to get additional information about the behaviour of the combined method. With these model orders different dynamic neural architectures were constructed, where in all cases one hidden layer of 30 neurons was applied. When Least Squares cost function was used the Levenberg-Marquardt training method was applied. With the EIV cost function the training was done using simple gradient descent search with adaptive training factor. The number of training cycles was 300 for both methods, and in both cases fast convergence could be observed during training.

Figs. 2 and 3 shows the summary of the results. On Fig. 2 we can see the sharp breakpoint at (0, 3) which is the true order.

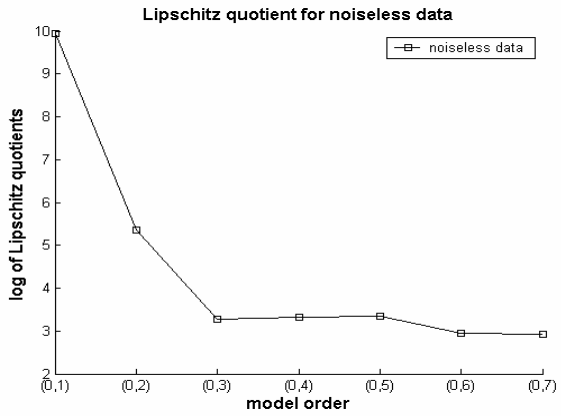


**Fig. 2 – Lipschitz quotients for the selected model order in noiseless.**

On Fig. 3 the results are presented for the four noisy cases. The graph belonging to the 0.01 noise variance is almost the same as in the noiseless case, so this small amount of additive noise has no significant effect. However, the next three cases do not have such sharp breakpoints and it is hard to decide which is the optimal order among (0, 2), (0, 3) and (0, 4). The higher orders (0, 5), (0, 6) and(0,7) cannot be good candidates as they are in the saturated range of the curve, while (0, 1) is definitely on the slope.

In our further experiments we focused on the three most likely orders. The tests are also done for other orders and it can be seen that if the model order is outside the

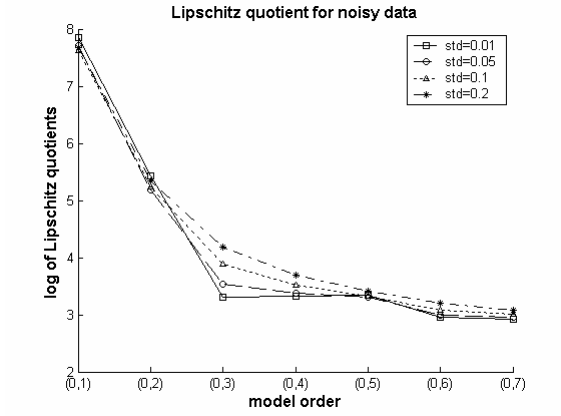


**Fig. 3 – Lipschitz quotients for the selected model order in the tested noise cases.**

most likely range, than training with EIV does not reduce the effect of noise in the input data, instead it may distort the input data further. The reason is that we try to adjust the data points to a too simple or a too complex mapping function. It is necessary to get the smoothest possible mapping before we start to use EIV while in Eq. (8) we use the slopes of the mapping to modify the input points. If the slope is too large then the changing of the input data will also be unnecessarily large (it leads to overtraining). If the mapping is smooth and the order is small, the slope is also small but in Eq. (8) we multiply the slope with the sample's error and this is large at small orders (The network cannot approximate the data correctly) and the changing of the input data is large again.

EIV utilises the variances of the input and output data. It can happen, that there are data samples with large noise variances, while the additive noise in other data samples is small. If there are data samples with very small variances then this sample's error is overweighted in the total error and the input update will be incorrect. It comes from the fact that EIV applies the reciprocal of the input and output variances, and these reciprocals can be extremely large. In our experiences this is the case when the variance is 0.01, so at further examinations we leaved this case out. If the noise is small the model order can be determined directly from the first run of the Lipschitz method, this is not a real weakness of the combined method. It is necessary to limit somehow the reciprocals to make the method applicable for small variances. Although a detailed analysis of this question will be the subject of a study in the near future, it is assumed that if some limit is introduced the robustness of the method for small noise variances can be improved.

During the evaluation of the experiments three quality measures are used. The first is the improvement in the Lipschitz criterion:

$$L_{imp} = \left(L_{final}^{(0,2)} - L_{final}^{(0,3)}\right) - \left(L_{init}^{(0,2)} - L_{init}^{(0,3)}\right) \qquad (11)$$

where $L_{init}^{(i,j)}$ is the Lipschitz quotient of the ($i$,$j$) model order at the initial Lipschitz estimation and $L_{final}^{(i,j)}$ is the Lipschitz quotient of the ($i$,$j$) model order at the final estimation. $L_{imp}$ measures the change in the graph of the two runs of Lipschitz algorithm. The higher $L_{imp}$ value means that the graph became sharper after training with EIV cost function. The second quality measure is the distance between the noisy and the modified input data:

$$D_{change} = \sqrt{\sum_{t=1}^{N} \left(x(t) - x^*(t)\right)^2} \qquad (12)$$

Where $x(t)$ denotes the noisy input data and $x^*(t)$ the modified input data. $D_{change}$ measures how much the EIV training modified the input data. The third measure is the following:

$$D_{correction} = \sqrt{\sum_{t=1}^{N} \left(x_{orig}(t) - x(t)\right)^2} - \sqrt{\sum_{t=1}^{N} \left(x_{orig}(t) - x^*(t)\right)^2} \qquad (13)$$

Where $x_{orig}(t)$ is the original noiseless input data. $D_{correction}$ measures the quality of the change on the input sample data. If $D_{correction}$ is high it means that the modified input data is closer to the original input data then the noisy one. The main problem with $D_{correction}$ is that it can be calculated only if we have the original noiseless input data, which is rare in real-life applications. Its importance is that we can show correlation between $L_{imp}$ and $D_{correction}$ therefore we can conclude from the measurable $L_{imp}$ to the value of $D_{correction}$ and the overall performance of the combined method.

On Fig. 4 we compare the results of the combined method for different noisy cases by the measures $L_{imp}$, $D_{change}$ and $D_{correction}$. We select the measurements with the same noise levels, for example with variation 0.05. We group these measurements into triplets, one measurement from the selected model orders: (0,2), (0,3), (0,4). We calculate a measure (for example $L_{imp}$) and select the model order with the highest measure value for every triplet (this is the winner). Than we count how many times a selected model order was a winner. The number of winners for every measure and every noise level can be seen on a histogram (for noise level 0.05 and measure $L_{imp}$ this is the top-left histogram). On Fig. 5 this process is done by repetition number and not by noise level.

To evaluate the histograms we can compare the winners of $L_{imp}$ and $D_{correction}$ on Fig (4). For noise level 0.05 and 0.1 it can be seen that the most frequent winner for both measures is the true model order. This points to the fact that the higher $L_{imp}$ is due to the higher $D_{correction}$, the combined method modifies the input data in the right way. The combined method helps to get better estimates for the model order (if $L_{imp}$ is higher the Lipschitz method identifies the model order sharper) and as a secondary effect we can identify whether the EIV corrects the samples or not (if $L_{imp}$ is not the highest than $D_{correction}$ is also not the highest). If the second run of the Lipschitz method does not improve than the change of the input data does not reduce the noise. The histograms for $D_{change}$ (middle column)

shows, that the change in the sample data is not significant versus the model order.

On Fig. 5 there are the results of the combined method for different measurement repetition number. The three rows are according to the different repetition numbers: 3, 5, 10. The figure shows that the combined method works well for *5* and *10* repetitions (the true model order is the winner at the histograms of $L_{imp}$ and $D_{correction}$).

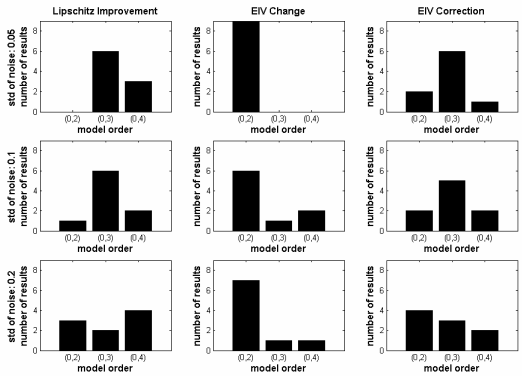


**Fig. 4 – Histograms of the results at different noise levels.**

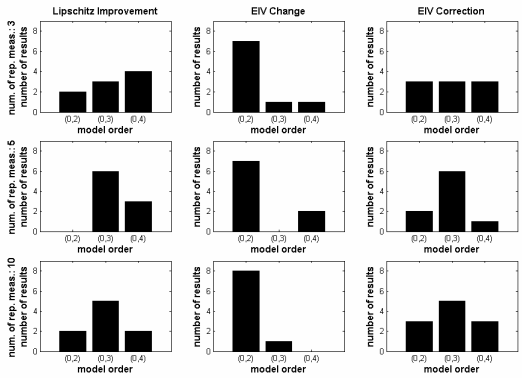


**Fig. 5 – Histograms of the results at different measurement repetition numbers.**

These amounts of measurements are enough to get "good" estimates of the noise variances. Repeating the measurements only 3 times leads to no significant results. The number of measurements in this case is too small to get a "good" estimate of the noise variance. The middle column shows, that $D_{change}$ is independent from the repetition number, also from noise variance. We can also determine as it is shown in Fig. 4, that the improvement of the Lipschitz method and the correction of the EIV are correlated.

## 7. CONCLUSIONS

In this paper a combined method to estimate the order of the model of a dynamic system was presented. The Lipschitz method, which is capable to estimate the model order directly from the measurement data, was combined with the EIV cost function that changes the training sequence during the estimation of the model's parameters. This effect was evaluated with a second run of the Lipschitz method. Comparing the two runs, the true model order can be determined and as a secondary effect we can determine whether the EIV corrects the noisy data or not. In our test we examined the robustness of the method against noise and the repetition number. It was found that small variances could cause instability of the method so some bound of the variance should be introduced. The detailed analysis of this possibility will be the subject of further study in the near future. Also an important next step will be to test the results on more complex real-life problems.

## 8. REFERENCES

*[1] K. Hornik, M. Stinchcombe, H. White. Multilayer Feedforward Networks are Universal Approximators, Neural Networks Vol. 2 (1989). pp. 359-366.*

*[2] J. Park, I. W. Sandberg. Approximation and Radial-Basis-Function Networks, Neural Computation, Vol. 5 No. 2 (1993). pp. 305-316.*

*[3] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B.Delyon, P.-Y. Glorennec, H. Hjalmarsson, A.Juditsky. Non-linear Black-box Modeling in System Identification: A Unified Overview, Automatica, 31 1691-1724 (1995).*

*[4] H. Akaike. A New Look at the Statistical Model Identification, IEEE Transaction on Aut. Control Vol. AC-19, No. 6 (1974). pp.716-723.*

*[5] J. Rissanen. Estimation of Structure by Minimum Description Length, Circuits, Systems and Signal Processing, special issue on Rational Approximations, Vol. 1, No. 3-4 (1982). pp. 395-406.*

*[6] H. Akaike. Statistical Predictor Identification, Ann. Istitute of Statistical Mathematics, Vol. 22 (1970). pp. 203-217.*

*[7] X. He, H Asada. A New Method for Identifying Orders of Input-output Models for Nonlinear Dynamic Systems. Proceedings of the American Control Conference, San Francisco, California June 1993, pp. 2520-2523.*

*[8] G. Vandersteen. Identification of Linear and Nonlinear Systems in an Errors-In-Variables Least Squares and Total Least Squares Framework. PhD thesis, Vrije Universiteit Brussel, Belgium April 1997.*

*[9] J. Van Gorp, J. Schoukens, R. Pintelon. The Errors-In-Variables Cost Function for Learning Neural Networks with Noisy Inputs, ANNIE 1998, Intelligent Engineering Systems Through*

*Artificial Neural Networks, Vol. 8 (1998). pp. 141-146.*

---

***Laszlo Sragner** received the M.S. degree in software engineering from the Technical University of Budapest (TUB) in 2002. He is currently pursuing the Ph.D. degree at the TUB at the Department of Measurement and Information Systems. His main research interests are non-linear system identification and hybrid intelligent systems.*

***Dr. Gabor Horvath** received the degree of engineer in 1970, the degree of doctor in 1987 all from the Technical University of Budapest (TUB). He is presently an associate professor at the TUB at Department of Measurement and Information Systems. The prime factors of his research is in the field of non-linear signal processing and theory of neural networks.*