Improvement of nonlinear dynamic models containing neural networks based on domain partitioning

Jörg Schultz, Volker Krebs
Institut für Regelungs- und Steuerungssysteme
Universität Karlsruhe
Kaiserstr. 12, 76128 Karlsruhe
Germany
schultz@irsserver.etec.uni-karlsruhe.de

Keywords: nonlinear systems, system identification, neural networks, dynamic error, domain partitioning

Abstract

One important objective in control theory is the identification of nonlinear dynamic processes. Recently, neural networks, which are well known as special types of approximators, have been used more and more as models for nonlinear systems. In order to obtain best fitting neural networks, in this paper a partitioning of the data space is proposed which is specified in particular by an estimation of the lattice density for appropriate data selection. This yields a strategy to diminish the dynamic error between system and model. It can be accomplished by adding well-suited process data for some detected significant differences between the behaviour of the model and the process.

1 Identification of the static part of the model

By introducing the time delay operator

$$\underline{T}_{n_1,n_2}(w_k) = \begin{bmatrix} w_{k-n_1} \\ w_{k-n_1-1} \\ \vdots \\ w_{k-n_2} \end{bmatrix}^T, n_2 \geq n_1 \geq 0 \quad (1)$$

(10) can be written as

$$\hat{y}_{k+1} = \hat{f}\left(\left[\underline{T}_{0,n_y}(\hat{y}_k), \underline{T}_{0,n_u}(u_k)\right]; \underline{p}\right).$$

The model is split into two components, the static nonlinearity $\hat{f}(\cdot;\underline{p})$ and its dynamic component

$$\underline{I} = \underline{I}(\hat{y}_k, u_k) = \left[\underline{T}_{0,n_y}(\hat{y}_k), \underline{T}_{0,n_u}(u_k)\right]$$

as can be seen from figure 1. The model dynamics results from the feedback and the time delays. The tupels $(\underline{x}^k, y^k)$ with

$$\begin{aligned} \underline{x}^k &= \left[\underline{T}_{0,n_y}(\hat{y}_k), \underline{T}_{0,n_u}(u_k)\right] \\ y^k &= y_{k+1} \end{aligned} \quad (2)$$

and $k = 1, \ldots, N$, $\underline{x}^k \in \Re^{n_e}$, $n_e = n_y + n_u + 2$, $y^k \in \Re$ are built from measured data. N indicates the number of tupels.

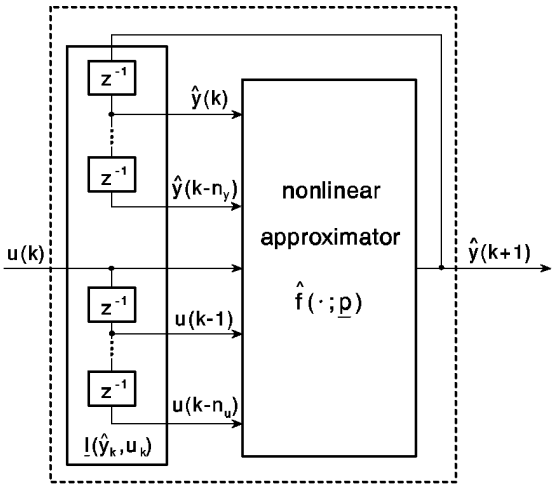

Figure 1: Nonlinear dynamic model which consists of a nonlinear approximator and a dynamic component.

The model parameters are determined by minimising the performance criterion

$$J = \sum_{k=1}^{N} (y^k - \hat{f}(\underline{x}^k; \underline{p}))^2 \quad (3)$$

with regard to the vector $\underline{p}$ using nonlinear optimisation methods [4]. Some classes of neural networks, e.g. multilayer perceptrons (MLP) [5], are appropriate types of nonlinear approximators. The parameter vector contains the weights and biases. These networks interpolate between the data points. The interpolation is called "generalisation" in neural network terms. Neural Networks

can be used to identify a process described by (9). Since a dynamic model (10) is to be identified, an optimisation method [2] that considers the dynamics should be used. However, the efforts of using a method like this are considerable because of slow convergence and high amount of calculations needed for this kind of algorithm. Thus, the static nonlinearity is approximated using process data (2). The dynamic model results by extending it with the time delay component.

1.1 Selection of training data

N is any given number and $\underline{x}^k \in \Re^{n_e}$ is any given location in the operating domain. In order to find a good system approximation, proper values for N and proper locations for $\underline{x}^k$ have to be determined. Since N signifies a large number of measured data and one cannot expect that $\underline{x}^k$ is well distributed in the input domain $\Re^{n_e}$, the problem to be solved is to determine a sufficient number $N_1 \leq N$ of well-suited $\underline{x}^k$, which are finally used for the approximation. The static nonlinear mapping $\hat{f}\left(\cdot;\underline{p}\right)$ describes a hypersurface in the domain with n_e+1 dimensions. In order to achieve good interpolation between the data points of $f\left(\cdot\right)$, it is obviously that one has to avoid large closed areas of the input domain where no $\underline{x}^k$ are found. Therefore the input domain is partitioned into hypercubes using a grid (see figure 2 with $n_e = 2$). Exactly one pattern consisting of $\underline{x}^k$ and the matching y^k from every cube is added to the set of training patterns.

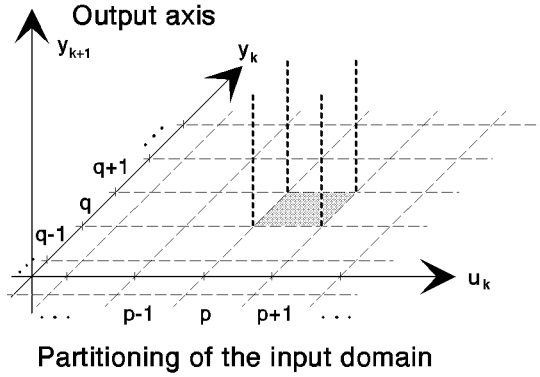

Figure 2: Domain partitioning with two variables of the input domain and one output variable.

The density of the lattice is derived by estimating the maximal error which is made within a cube. The n_e axes of the input domain are equidistantly divided with a lattice constant

$$\Delta d = \Delta d_{y_k} = \cdots = \Delta d_{y_{k-n_y}} = \Delta d_{u_k} = \cdots = \Delta d_{u_{k-n_u}} . \tag{4}$$

Introducing the vector

$$\begin{aligned} d\underline{I} &= \left[dy_k, \ldots, dy_{k-n_y}, du_k, \ldots, du_{k-n_u}\right]^T \\ &= \left[d\underline{T}_{0,n_y}\left(y_k\right), d\underline{T}_{0,n_u}\left(u_k\right)\right]^T \end{aligned}$$

the total differential of $f\left(\cdot\right)$

$$dy_{k+1} = \left[\frac{\partial}{\partial \underline{I}} f\left(\underline{I}\right)\Big|_{\tilde{\underline{I}}}\right]^T d\underline{I}$$

can be calculated at every $\tilde{\underline{I}} = \underline{I}\left(\tilde{y}_k, \tilde{u}_k\right)$ of the input domain. It has to be assumed that $f\left(\cdot\right)$ is differentiable in the whole operating domain. The tangent plane at $\tilde{\underline{I}} \in \Re^{n_e}$ is given by

$$t = \underline{m}^T \Delta \underline{I} \tag{5}$$

with variable

$$\Delta \underline{I} = \left[\Delta y_k, \ldots, \Delta y_{k-n_y}, \Delta u_k, \ldots, \Delta u_{k-n_u}\right]^T \tag{6}$$

and the gradient vector

$$\underline{m} = \frac{\partial}{\partial \underline{I}} f\left(\underline{I}\right)\Big|_{\tilde{\underline{I}}} .$$

Its norm

$$\|\underline{m}\| = \| \left[m_1, \ldots, m_{n_e}\right]^T \| = \sqrt{\sum_{i=1}^{n_e} m_i^2}$$

gives information about the slope of the tangent plane. Assuming that $\|\underline{m}\| \leq M_{max}$ holds in the operating domain of $f\left(\cdot\right)$, the estimation of the maximal error within a hypercube is given by

$$\|\varepsilon_y\| = \|\underline{m}\| \, \|\Delta \underline{I}\| \leq M_{max} \|\Delta \underline{I}\|$$

when linear interpolation based on the tangent surface (5) is used. Let $M_{max}\|\Delta \underline{I}\|$ be less equal than a limit $\varepsilon_{y,max} > 0$ that has to be specified. With (4) we get

$$\Delta \underline{I} = \left[\Delta d_{y_k}, \ldots, \Delta d_{y_{k-n_y}}, \Delta d_{u_k}, \ldots, \Delta d_{u_{k-n_u}}\right]^T , \tag{7}$$

hence

$$M_{max} \Delta d \sqrt{n_e} \leq \varepsilon_{y,max} ,$$

thus

$$\Delta d \leq \frac{\varepsilon_{y,max}}{M_{max}\sqrt{n_e}} \tag{8}$$

as an estimation for the necessary lattice constant Δd. Taking into account that the number of hypercubes grows exponentially with decreasing Δd , the worst-case- estimation (8) of Δd indicates an unnecessarily small lattice constant. A more practical approach is to consider the mean of the error in a hypercube. For those parts of the input domain where the lattice is not sufficiently dense, it is possible to change locally to a smaller lattice constant. The mean error is computed by

$$\overline{\|\varepsilon_y\|} = \frac{1}{H}\sum_{k=1}^{H} \|\varepsilon_{y_k}\| = \frac{1}{H}\sum_{k=1}^{H} \|\varepsilon_y\left(\underline{I}_k\right)\|$$

when the input domain is divided into H hypercubes. With $\|\varepsilon_y(\underline{I}_k)\| = \|\underline{m}(\underline{I}_k)\|\,\|\Delta\underline{I}\|$ it follows $\overline{\|\varepsilon_y(\underline{I}_k)\|} = \overline{\|\underline{m}\|}\,\|\Delta\underline{I}\|$. $\overline{\|\underline{m}\|}$ is the arithmetic mean of the norm of all H gradient vectors in the operating domain. Since $\overline{\|\Delta\underline{I}\|} = \|\Delta\underline{I}\|$, it follows

$$\Delta d = \frac{\varepsilon_{y,max}}{\overline{M}_{max}\sqrt{n_e}}.$$

To fill the hypercubes, process data (2) has to be scanned in such a way that, finally, there is one pattern in every hypercube of the operating domain. Thus, the training patterns are distributed evenly without gaps. After minimising the performance criterion (3) the model with the resulting parameters must hold the condition

$$\|\underline{\hat{m}}\| = \left\| \frac{\partial}{\partial \underline{I}} \hat{f}(\underline{I}) \bigg|_{\tilde{\underline{I}}} \right\| \leq M_{max}.$$

If there is no explicit information about $M_{max}, \overline{M}_{max}$, it has to be estimated. The tangent plane (5) motivates an estimation model

$$t = \underline{m}_{est}^T \Delta\underline{I}.$$

The gradient vector $\underline{m}_{est}$ is computed at every pattern $\underline{I}$ in the operating domain using a least-squares-method. To obtain a good estimation, only those patterns are used which lie in the adjacent hypercubes. The calculation has to be carried out with an appropriately small lattice constant. The smaller it is, the better is the estimation.

Example 1: The process taken from [2] described by the following difference equation

$$y_{k+1} = \frac{y_k y_{k-1}(y_k + 2.5)}{1 + y_k^2 + y_{k-1}^2} + u_k$$

is considered. The unforced system is to be identified. The operating domain for the output variable is chosen to $-1.0 \leq y_k \leq 1.0$ where it holds $M_{max} = 4.41$ and $\overline{M}_{max} = 2.17$. With the specification of

$$\varepsilon_{y,max} = 0.05|y_{max} - y_{min}|$$

(8) yields $\Delta d \leq 0.04$. The lattice constant is chosen to $\Delta d = 0.04$. An MLP with 30 neurons in one hidden layer and the sigmoid function as activation function is used. The training is carried out until the relative error sum of squares

$$\varepsilon_{rel} = \frac{\sum_k^{10000}(y_k - \hat{y}_k)^2}{\sum_k^{10000} y_k^2} = 9.9 \cdot 10^{-8}$$

is reached. Starting from 729 evenly distributed initial states, there is only one significant divergence detected between the behaviour of the model and the system. It is the unstable equilibrium point $[y_k, y_{k-1}] = [0.5, 0.5]$ of the process. In contrast to the system, which stays in this equilibrium point when no disturbances occur, the neural network leaves this point due to the remaining error after minimisation. The MLP has the unstable equilibrium point $[y_k, y_{k-1}] = [0.5057, 0.5057]$. Further trials with the same boundary conditions and lattice constants $\Delta d = 0.25$ and $\Delta d = 0.1$ were carried out. In these cases, considerably more initial states lead to significant divergences.

2 Introduction

Neural networks are applied more and more in control theory [1]. They are used to approximate sufficiently well continuous static nonlinearities [3]. Extending the static nonlinearity with a dynamic component considering feedback and time delays, they can be employed to identify nonlinear dynamic systems [2], in particular, for which the mathematical model cannot be derived analytically. Basically, the problem of selecting appropriate process data has to be solved. Therefore, the domain which is introduced by the input/output variables of the model and their delays needs to be partitioned sufficiently. The suitable density of the resulting lattice is evaluated and the model parameters are determined. In case the model does not satisfy all requirements, a procedure will be presented to improve the model behaviour. This is achieved by extending the identification data set with new relevant process data. The result is an iterative procedure for reducing the differences between the dynamic behaviour of model and process.

3 The nonlinear dynamic process

It is assumed that a nonlinear dynamic SISO process is to be identified. The process is BIBO-stable in the considered operating domain. Thus, one can measure process data where the functionality of the process according to a difference equation of the form

$$y_{k+1} = f(y_k, \ldots, y_{k-n_y}, u_k, \ldots, u_{k-n_u}) \qquad (9)$$

is found. The nonlinear mapping $f(\cdot)$ is unknown and assumed to belong to the class of time-invariant, bounded, continuous and differentiable operators. The number of time delays which influences the process output y_{k+1} is indicated as n_y (for the output which is fed back) and n_u (for the input variable u_k). A model deduction is compulsory if system analysis and control synthesis based on a model are to be accomplished by simulation. The model is introduced with the following difference equation

$$\hat{y}_{k+1} = \hat{f}(\hat{y}_k, \ldots, \hat{y}_{k-n_y}, u_k, \ldots, u_{k-n_u}; \underline{p}). \qquad (10)$$

The vector $\underline{p}$ consists of the model parameters. They have to be determined in such a way that (10) approximates the input/output behaviour of (9) as well as possible.

4 The dynamic error of the model

After specifying the lattice constant, selecting the data and minimising the criterion (3) until a remaining error, the nonlinearity $\hat{f}\left(\cdot;\underline{p}\right)$ is determined. The dynamic model is set up with the feedback and the time delay elements. When model and system start from the same initial state

$$\left[\underline{T}_{0,n_y}(y_k),\underline{T}_{0,n_u}(u_k)\right]$$

and an input sequence (u_k) is used, a dynamic error sequence results as follows:

$$\begin{aligned}
\varepsilon_k &= y_{k+1}-\hat{y}_{k+1}\\
&= f\left(\left[\underline{T}_{0,n_y}(y_k),\underline{T}_{0,n_u}(u_k)\right]\right)\\
&\quad -\hat{f}\left(\left[\underline{T}_{0,n_y}(y_k),\underline{T}_{0,n_u}(u_k)\right];\underline{p}\right),\\
\varepsilon_{k+1} &= y_{k+2}-\hat{y}_{k+2}\\
&= f\left(\left[\underline{T}_{0,n_y}(y_{k+1}),\underline{T}_{0,n_u}(u_{k+1})\right]\right)\\
&\quad -\hat{f}\left(\left[\hat{y}_{k+1},\underline{T}_{0,n_y-1}(y_k),\underline{T}_{0,n_u}(u_{k+1})\right];\underline{p}\right),\\
&\vdots\\
\varepsilon_{k+n_y+1} &= y_{k+n_y+2}-\hat{y}_{k+n_y+2}\\
&= f\left(\left[\underline{T}_{0,n_y}(y_{k+n_y+1}),\underline{T}_{0,n_u}(u_{k+n_y+1})\right]\right)\\
&\quad -\hat{f}\left(\left[\underline{T}_{0,n_y}(\hat{y}_{k+n_y+1}),\underline{T}_{0,n_u}(u_{k+n_y+1})\right];\underline{p}\right),\\
&\vdots
\end{aligned}$$

The objective is to reduce the absolute value of the sequence elements $\varepsilon_j, j\geq k$ to a specified limit. Since the nonlinear mapping $f(\cdot)$ is not available, the analysis and the improval of the dynamic model behaviour is performed by using identification data.

5 Analysis of the dynamic model and measures of its improval

Concerning the model validation, model and system, which are initialised with the same states, are fed with an input sequence (u_k). An output sequence of the model $(\hat{y}_k)$ results which is compared with the process sequence (y_k). In case the error between system output and model output exceeds an unacceptable limit, either the model has to be rejected or measures have to be taken to improve the model behaviour.

To analyse the origin of the model error, the space spanned by the n_e input variables and the output variable has to be considered. Applying the time delay operator (1), the n_e-dimensional domain vector

$$\underline{v}_{k+1}=\left[T_{0,n_y}(y_k),T_{0,n_u}(u_k)\right]$$

and its output y_{k+1} (for the system) as well as

$$\hat{\underline{v}}_{k+1}=\left[T_{0,n_y}(\hat{y}_k),T_{0,n_u}(u_k)\right]$$

and its output $\hat{y}_{k+1}$ (for the model) are built.

The approximation error cannot be reduced to zero, therefore, the output of the system y_{k+1} and of the model $\hat{y}_{k+1}$ are in general different. Due to dynamic error backprogation as explained in section 4, it is not sufficient only to reduce the static approximation error.

A measure for good dynamic behaviour of the model is derived from the difference between the domain vectors of model and system

$$\|\Delta\underline{v}_{k+1}\|=\|\underline{v}_{k+1}-\hat{\underline{v}}_{k+1}\|. \tag{11}$$

A difference will be considered as significant if it exceeds a limit $\Delta v_{min}>0$

$$\|\Delta\underline{v}_{k+1}\|>\Delta v_{min}. \tag{12}$$

From (y_k) and $(\hat{y}_k)$ a sequence $(\Delta\underline{v}_k)$ results. The duration of a coherent sequence of significant differences

$$\begin{aligned}
&\Delta\underline{v}_i,\ldots,\Delta\underline{v}_{i+m};\|\Delta\underline{v}_j\|>\Delta v_{min},j=i,\ldots,i+m\\
&\quad and\ \ \|\Delta\underline{v}_j\|\leq\Delta v_{min},j=i-1,i+m+1
\end{aligned} \tag{13}$$

is the period m from the appearance of the first $\|\Delta\underline{v}_i\|>\Delta v_{min}$ until the last $\|\Delta\underline{v}_{i+m}\|>\Delta v_{min}$. Apart from condition (12), a minimal duration k_{min} can be introduced, so that errors with $\|\Delta\underline{v}_j\|>\Delta v_{min}$ for a short coherent sequence of significant errors are not considered. A significant error $\|\Delta\underline{v}_{k+1}\|>\Delta v_{min}$ at sample $k+1$ is the result of approximation and interpolation errors of its preceding model outputs for which (12) does not hold. The approximation and interpolation properties of the model for $k_p=k_{p,min}>0$ preceding domain tupels has to be improved. This is reached by adding new tupels $(\underline{v}_i,y_i), i=1,\ldots,N_{add}$ to the training set. These have to lie inside a hypersphere $\|\underline{v}_i-\underline{v}_j\|\leq\Delta v_{min}$ around the vectors $\underline{v}_j, j=k-k_p+1,\ldots,k$. The number of added tupels N_{add} has to be bounded so that no part of the input domain is overemphasized. With the grid introduced in section (1.1), only a maximal number of new tupels is allowed for every hypercube. Using the vector $\underline{v}_j$, it is guaranteed that tupels are only added in those parts of the input domain where it is necessary. Therefore, the number of learning patterns does not increase too much. This is advantageous with regard to training time.

Further network training leads to a model with better properties of approximation and interpolation. The training with the extended pattern set does not take too long, since one starts from an already trained network. The procedure of generating the input vector sequences, examining the differences and, finally, extending the training set with new patterns can be accomplished iteratively as shown in figure 3. It is repeated until a certain model quality is achieved. The following example

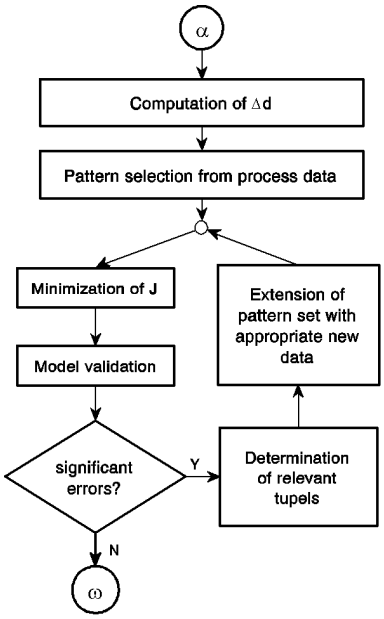

Figure 3: Flow chart of the iterative procedure.

illustrates the procedure.

Example 2: A continuous stirred tank reactor is to be identified. The inaccessible continuous time model [6]

$$\dot{\underline{x}} = \begin{bmatrix} -0.957 & a_{12}(\underline{x}) \\ -0.323 & a_{22}(\underline{x}) \end{bmatrix} \underline{x} + \begin{bmatrix} 0 \\ 1.548 \end{bmatrix} u$$

with

$$a_{12}(\underline{x}) = \begin{cases} b_1(\underline{x}) \left(e^{b(x_2)} - 6.568 \cdot 10^{-15} \right) & : \quad x_2 \neq 0 \\ 21.449 \, (0.279 - x_1) & : \quad x_2 = 0 \end{cases}$$

$$a_{22}(\underline{x}) = 0.468 \, a_{12}(\underline{x}) - 1.815$$

$$b_1(\underline{x}) = \frac{1.05 \cdot 10^{14} \, (0.279 - x_1)}{x_2} \; ; \; b(x_2) = \frac{-34.289}{1.05 + x_2}$$

represents the process. The coolant temperature is the control variable u. Reactor temperature and concentration of the input flow in the tank are state variables. In the following example, the model for the concentration is built.

Intentionally, the initial training patterns are unevenly distributed on the input domain to emphasise the effects of the measures to improve the behaviour of the model. The model is formulated with $n_y = 1$ and $n_u = 1$. The nonlinear model with the neural network is described by

$$\hat{y}_{k+1} = \hat{f} \left(y_k, \hat{y}_{k-1}, u_k, u_{k-1}; \underline{p} \right).$$

The operating intervals are $[-0.1, 0.1]$ and $[-0.5, 0.5]$ for the control variable and the output variable respectively. The axes are divided with $\Delta y_k = \Delta y_{k-1} = 0.02$ and $\Delta u_k = \Delta u_{k-1} = 0.002$. The number of neurons in one hidden layer in the MLP is chosen to 25. The standard sigmoid function is used as activation function. Input/output data of the process starting from different initial states using 4 different input sequences (u_k) is measured. The initial training set consisting of 792 patterns signifies that the four-dimensional input domain is not very dense. Apart from this, a set of about 4000 patterns is built from process data. During the procedure, patterns are taken from this data pool to extend the training set. The minimal difference $\Delta v_{min} = 0.071$ in the following examination. A minimal period k_{min} is not specified.

The error sum of squares $SSE_{abs} = \sum_k (y_k - \hat{y}_k)^2 = \sum_k (y_k - \hat{f}(\underline{x}^k; \underline{p}))^2$ and the relative error sum of squares

$$SSE_{rel} = \frac{\sum_k (y_k - \hat{y}_k)^2}{\sum_k y_k^2} = \frac{\sum_k (y_k - \hat{f}(\underline{x}^k; \underline{p}))^2}{\sum_k y_k^2}$$

are minimised by net training to $SSE_{abs} = 0.87 \cdot 10^{-5}$ and $SSE_{rel} = 1.3 \cdot 10^{-6}$ after 300 epochs. The validation with an input sequence (u_k), $u_k \in [-0.1, 0.1]$ gives the output of model and process as shown in fig. 4.

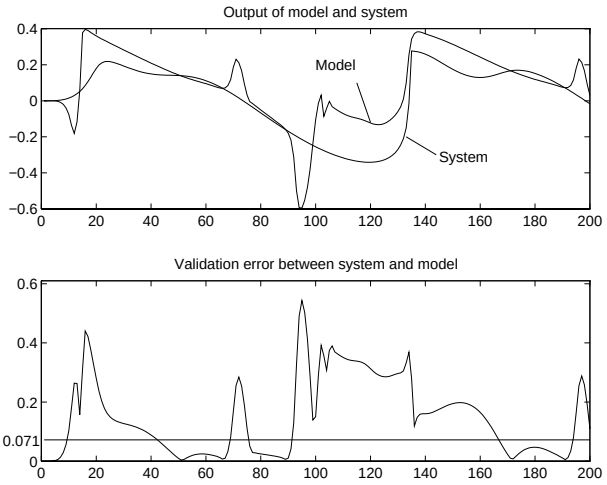

Figure 4: Curves of system and model after 300 epochs of training with 792 patterns. The lower plot shows the difference $\|\underline{v}_k - \underline{\hat{v}}_k\|$.

The dynamic error sum of squares

$$DSSE_{abs} = \sum_k \left(y_k - \hat{f} \left(\left[T_{0,n_y}(\hat{y}_k), T_{0,n_u}(u_k) \right], \underline{p} \right) \right)^2$$

and the relative dynamic error sum of squares

$$DSSE_{rel} = \frac{\sum_k \left(y_k - \hat{f} \left(\left[T_{0,n_y}(\hat{y}_k), T_{0,n_u}(u_k) \right]; \underline{p} \right) \right)^2}{\sum_k y_k^2}$$

for this net are $DSSE_{abs} = 3.94$ and $DSSE_{rel} = 0.583$. After examining the sequence of differences of domain vectors $(\underline{v}_k), (\underline{\hat{v}}_k)$ of system and model respectively, 4 significant errors remain at $k = 9, 68, 91, 193$ (see fig. 4).

New data is added to the training set. The number of patterns rises to 826 with $k_p = 3$. For the difference

at $k = 91$ only one tupel ist added. In the data pool no further tupels are available whose domain vectors lie close to $\underline{v}_{88}, \underline{v}_{89}, \underline{v}_{90}$. A new training of 200 epochs is performed until a similar error $SSE_{abs} = 1.03 \cdot 10^{-5}$ and relative error $SSE_{rel} = 1.52 \cdot 10^{-6}$ is reached. This leads to a model the output of which is presented in figure 5.

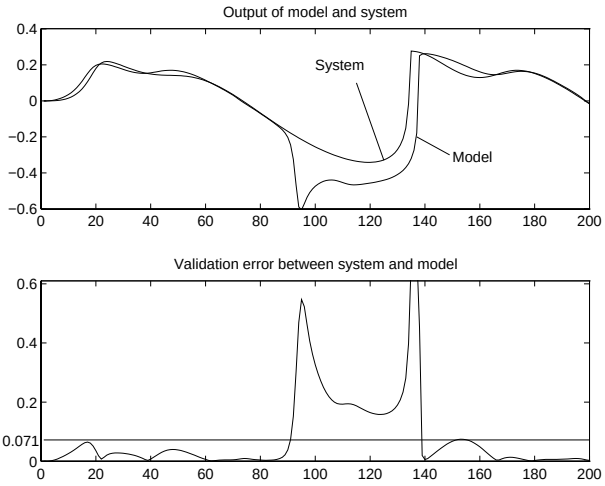

Figure 5: Curves of system and model after further 200 epochs with 826 patterns. The lower plot shows the difference $\|\underline{v}_k - \hat{\underline{v}}_k\|$.

Comparing the output of model and process, the result is a dynamic error $DSSE_{abs} = 1.912$ and $DSSE_{rel} = 0.283$. Thus, a reduction of the dynamic error is achieved. Only one significant difference remains at the sample $k = 91$. No matching data was found in the data pool for the $k_p = 3$ precedings domain vectors $\underline{v}_{88}, \underline{v}_{89}, \underline{v}_{90}$. Note that the extension with data which is located in relevant areas leads to an improval, so that no differences occur anymore for these tupels where new data is added ($k = 9, 68, 193$).

It has to be mentioned that this result cannot be achieved only by continuing the training of the first net. A further training of 300 epochs of this net, finally, yields $SSE_{abs} = 6.34 \cdot 10^{-6}$ and $SSE_{rel} = 9.3 \cdot 10^{-7}$ that shows significant differences at similar samples $k = 14, 26, 68, 91, 194$ (see fig. 6).

6 Conclusion

For the identification of nonlinear dynamic systems with neural networks, it is necessary to select appropriate process data. Therefore, a method based on domain partitioning is proposed in this paper. The derived lattice constant, which leads to a sufficient data density, may imply high cost of computation. On the other hand, less training data may lead to a large dynamic error. With the proposed measures of extending the pattern set appropriately, the dynamic error could be reduced noticeably. An iterative procedure to improve the model behaviour was applied to identify a continuous stirred tank reactor using

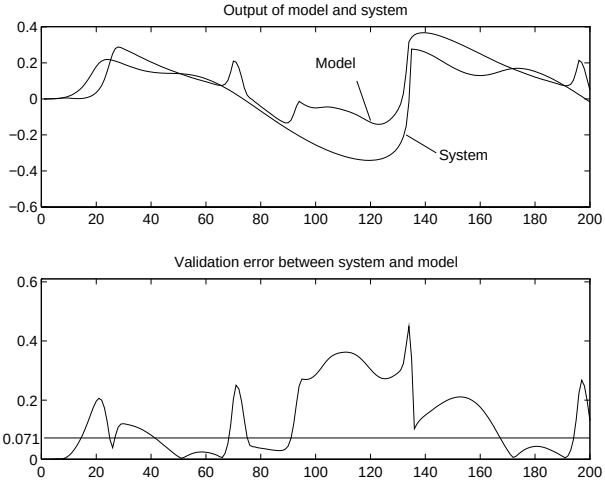

Figure 6: Curves of system and model after 600 epochs of training with 792 patterns. The lower plot shows the difference $\|\underline{v}_k - \hat{\underline{v}}_k\|$.

a multilayer perceptron as model.

References

[1] Hunt, K.J., Sbarboro, D.: "Neural Networks for Control Systems - A Survey", Automatica, 28(6), pp.1083-1112, 1992.

[2] Narendra, K. S., Parthasarathy, K.: "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Trans. on Neural Networks, 1(1), pp. 4-27, March 1990.

[3] Hornik, K.: "Approximation capabilities of multilayer feedforward networks", Neural Networks, vol. 4, pp. 251-257, 1991.

[4] Press, W. H.: "Numerical Recipes in C", Cambridge University Press, Second Edition, 1992.

[5] Rumelhart, D. E., McClelland, J. L.: "Parallel distributed processing", Volume 1&2, MIT Press, Cambridge, 1986.

[6] Föllinger, O.: "Nichtlineare Regelungssysteme I,II", R.Oldenbourg, München , 7. Edition, 1993.