Inversion of Fuzzy Neural Networks for the Reduction of Noise in the Control Loop for Automotive Applications

Nentwig, M.; Mercorelli, Paolo

Published in:
JAMRIS, Journal of Automation, Mobile Robotics & Intelligent Systems

Publication date:
2009

Document Version
Publisher's PDF, also known as Version of record

Inversion of Fuzzy Neural Networks for the Reduction of Noise in the Control Loop for Automotive Applications

Mirko Nentwig, Paolo Mercorelli

Abstract:

A robust throttle valve control has been an attractive problem since throttle by wire systems were established in the mid-nineties. Control strategies often use a feed-forward controller which use an inverse model; however, mathematical model inversions imply a high order of differentiation of the state variables resulting in noise effects. In general, neural networks are a very effective and popular tool for modelling. The inversion of a neural network makes it possible to use these networks in control problem schemes. This paper presents a control strategy based upon an inversion of a feed-forward trained local linear model tree. The local linear model tree is realized through a fuzzy neural network. Simulated results from real data measurements are presented, and two control loops are explicitly compared.

Keywords: *neural networks, fuzzy control, inversion of neural networks, automotive control, noise reduction.*

1. Introduction

The automobile industry often models its engines using characteristic diagrams, or more specifically, engine operating maps. These models require a large amount of measured data acquired with advanced instrumentation. Alternatively, physics-based models may be used, but these are very complex and must be simplified for use, which degrades the model. Neural networks are another option for modelling complex systems. They are relatively straightforward systems but they may be appropriate even for highly complex modelled problems. The purpose of our work is to show that neural networks can be applied successfully, even in the presence of noise. We apply an inverse local linear model tree using a fuzzy neural network to a control loop, in the presence of noise. The considered system is the throttle valve shown in Fig. 1, which is displayed along with the parts of the internal combustion engine. The right side of Fig. 1 shows an enlargement of the throttle valve with its most important parameters. C_1 and C_2 are the mass flow rate for the input and output, respectively. Also, T_1, P_1 and T_2, P_2 represent the input and output temperature and pressure. A_1 is the total surface area of the plate, and $A_D = A_1 \cos(\gamma)$.

A robust throttle valve control has been an attractive problem since throttle by wire systems were established in the mid-nineties. An already tested technology is currently available, and recent advanced studies have appeared, [1] and [13]. Mercorelli [8] presented a controller based on inversion using a physical model approach. In particular, the author adopted the following model:

$$\frac{\partial i(t)}{\partial t} = \frac{R_{Coil}}{L_{Coil}} i(t) + \frac{u_{in}(t) - C_m \omega(t)}{L_{Coil}} \quad (1)$$

$$\frac{\partial \gamma(t)}{\partial t} = \omega(t) \quad (2)$$

$$\frac{\partial \omega(t)}{\partial t} = g_r \frac{C_m}{J} i(t) + g_r \frac{-k_r \omega(t) - T_{k_{pre}} - k_f(\gamma)\gamma(t)}{J}, \quad (3)$$

in which equation (1) represents the electrical system of the actuator, and equations (2) and (3) describe the mechanical behaviour of the actuator. The coil current $i(t)$, the angular position γ, and the angular velocity ω are the state variables, and $u_{in}(t)$ is the input voltage. R_{Coil} and L_{Coil} are the resistance and the inductance of the coil windings.

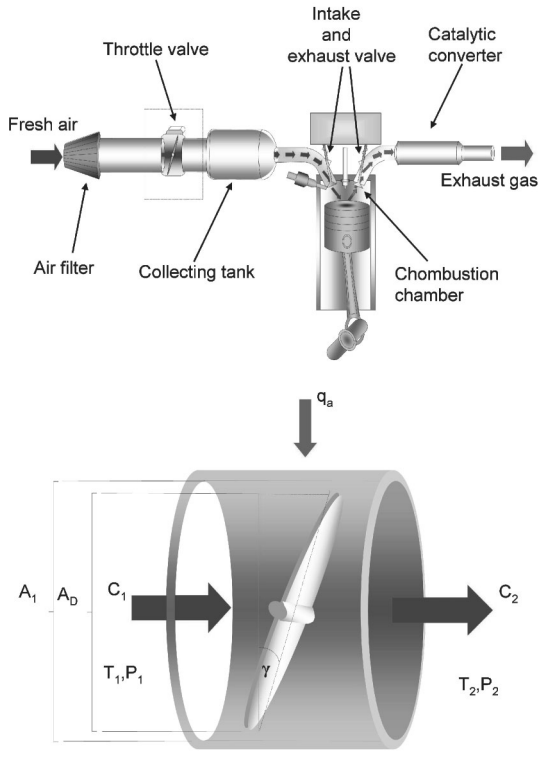

Fig. 1. Top: Overview. Bottom: Schematic structure of throttle valve.

$C_m\omega$ is normally called induced voltage, C_m is the constant of the motor, and J is the moment of inertia. The gear parameter g_r indicates the ratio of the teeth. With this approach, the backlash effect does not generate a stationary error. The parameters $k_r\omega$ and $k_f(\gamma)\gamma$ represent the viscous friction torque and the total spring

torque, respectively. $T_{k_{pre}}$ is the pretension torque of the spring, which can be considered as a disturbance. It should be noted that in our case, $k_f(\gamma)\gamma$ is a non-linear function of the angular position.

In particular, the following expression

$$T_L = C_m i(t)$$

describes the Lorentz torque generated by the actuator. Mercorelli [8] showed that the adopted model is a flat model and that

$$u_{in}(t) = \frac{-k_r L_{Coil}(C_m i(t) - k_f(y)y) + k_r^2 L_{Coil}\,\dot{y} C_m J}{C_m J} - \frac{J(C_m R_{Coil} i(t) + C_m^2 \dot{y} + k_f(y) L_{Coil}\dot{y} + J L_{Coil}\ddot{y})}{C_m J} \quad (4)$$

is the *inverse system* with $y = \gamma$. Because of the noise which can affect the signal from the foot pedal, a feed forward inverse controller may generate spikes and low tracking performances. From (4), it should be noted that mathematical inversions of models imply a high order of differentiation of the state variables, and consequently noise effects. Mercorelli [8] avoided this noise effect by developing an approximated proportional derivative (PD) regulator, in which the D-part was replaced with a special algorithm. Since it is always present a structural inexact description of the model with imperfect inversion, and external disturbances were not modelled, it was necessary for the control loop to contain a feedback structure.

2. Model inversion

The inversion problem in neural networks has attracted many researchers and mathematicians. This is a difficult problem, which involves the inversion of the non-linear membership functions. Fig. 2 presents a schematic structure of a possible control system. This procedure applies the LOLIMOT algorithm [9], which is based upon Neural-Fuzzy models of Takigi Sugeno type. During the execution of this algorithm, a "divide and conquer strategy" is applied to the modelling problem, so that the major problem is split into smaller ones. The basic network structure of a local linear neural fuzzy model is depicted in Fig. 3. Every neuron consists of a local linear model (LLM) and a validity function Φ defining the validity of the LLM within the input space. The local linear model output is defined by:

$$\hat{y}_i = w_{i0} + w_{i1} u_1 + w_{i2} u_2 + \ldots + w_{ip} u_p,$$

where w_{ij} is the LLM parameter at each neuron i.

If the validity functions are chosen as normalized Gaussians, then it follows:

$$\sum_{i=1}^{M} \Phi_i(\underline{u}) = 1 \text{ and } \Phi_i = \frac{\mu_i(\underline{u})}{\sum_{i=1}^{M} \mu_j(\underline{u})}$$

where the membership function μ is

$$\mu_i(\underline{u}) = exp\left(-\frac{1}{2}\left(\frac{(u_1 - c_{i1})^2}{\sigma_{i1}^2}\right)\right) + exp\left(-\frac{1}{2}\left(\frac{(u_2 - c_{i2})^2}{\sigma_{i2}^2}\right)\right) + \ldots + exp\left(-\frac{1}{2}\left(\frac{(u_p - c_{ip})^2}{\sigma_{ip}^2}\right)\right).$$

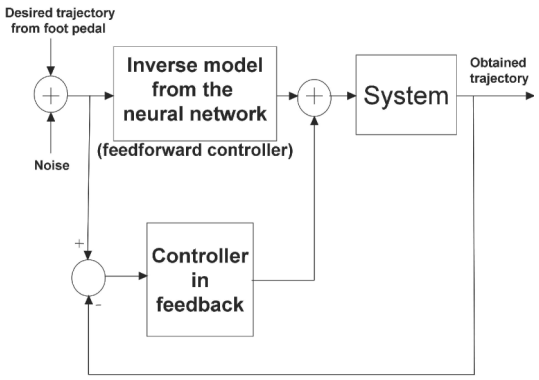

Fig. 2. Complete control scheme.

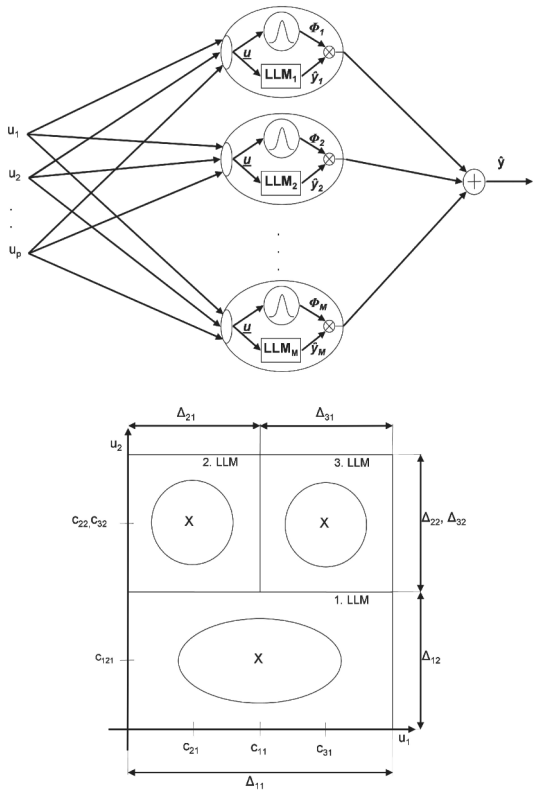

Fig. 3. Top: Network structure of local linear neural fuzzy model. Bottom: Partition of the input space by validity functions.

To achieve an inversion, we develop an algorithm which allows us to obtain the required model input u_r, depending on the desired model output y and the other model inputs $\underline{u}$.

Fink and Toepfer [4] offer some strategies in their analysis of the inversion of non-linear models:

- ***Inverse access by numerical inversion***
  Only one model of the non-linear function is created and used for standard and inverse access. The inverse access equals a numerical inversion and requires the application of optimization methods to determine the input for the requested output.

- ***Data driven generation of an inverse model***
  A model for inverse access is created in addition to the model for standard access.
- ***Analytical inversion of models***
  A direct inversion of the forward trained model is applied. Hence, it is an advantage to use model architectures, which allow the direct calculation of the inverse model using its own parameters.

The developed algorithm applies an analytical/numerical inversion of a given local linear model structure, and is explained below. The following constraints are required to set up the algorithm:

1. An existing forward trained local linear model tree of the process is available.
2. The expected model output y is known.
3. There are existing input values for that inputs upon which y is dependent.

2.1. The Validity Functions Issue

Consider the model output function $\hat{y}$, with M local linear model $\underline{u} = [u_1, ..., u_p]$ inputs

$$\hat{y}_i = \sum_{i=1}^{M} (w_{i0} + w_{i1}u_1 + + w_{i2}u_2 + ... + w_{ip}u_p)\Phi_i(\underline{u}), \qquad (6)$$

the validity function

$$\Phi_i = \frac{\mu_i(\underline{u})}{\sum_{i=1}^{M} \mu_j(\underline{u})}$$

and the membership function

$$\mu_i(\underline{u}) = exp\left(-\frac{1}{2}\left(\frac{(u_1 - c_{i1})^2}{\sigma_{i1}^2}\right)\right) + exp\left(-\frac{1}{2}\left(\frac{(u_2 - c_{i2})^2}{\sigma_{i2}^2}\right)\right)$$
$$+ ... + exp\left(-\frac{1}{2}\left(\frac{(u_p - c_{ip})^2}{\sigma_{ip}^2}\right)\right). \qquad (7)$$

A difficulty is that, due to the exponential quadratic nonlinearity, the model is not invertible. Hence, it is necessary to convert the functions into a linear type, as shown in Fig. 4. The membership function is split into a spline function that consists of the linear functions.

$$\mu_{ir} = \begin{cases} \frac{k_\sigma}{1.6 \cdot \sigma_r}(u_r - c) + 1 & -\frac{1.6 \cdot \sigma_r}{k_\sigma} + c \le u_r \le c \\ -\frac{k_\sigma}{1.6 \cdot \sigma_r}(u_r + c) + 1 & +c \le u_r \le \frac{1.6 \cdot \sigma_r}{k_\sigma} + c. \end{cases} \qquad (8)$$

Using the function defined in equation (8), it is now possible to invert the local linear model tree.

2.2. Algorithm for the Inversion of the LLM

We apply the following algorithm to invert the model:

1. *Calculate the LLMs represented in equation (6), omitting the required input variable u_r.*
   That is, calculate the LLMs with the available input data until there only remains a linear equation depending on one input, e.g., $y_i = w_r \cdot u_r + u_{calc}$. To be more comprehensible, if $\hat{y}_i = w_{i0} + w_{i1}u_1 + w_{i2}u_2 + w_{i3}u_3$ and u_{i1} is required, then $y_i = w_{i1}\, u_{i1} + u_{calc}$, with $w_r u_r = w_{i1}u_{i1}$ and $u_{calc} = w_{i0} + w_{i2}u_{i2} + w_{i3}u_{i3}$.

2. *Calculate the membership functions represented in equation (7), omitting the required input variable u_r.*
   The membership function of the LLMs is calculated with the available input data to the extent possible. Since the non-linear term depending on u_r is omitted, the membership function is a constant number. To be more precise,

$$\mu_i(\underline{u}) = exp\left(-\frac{1}{2}\left(\frac{(u_2 - c_{i2})^2}{\sigma_{i2}^2}\right)\right) + ...$$
$$+ exp\left(-\frac{1}{2}\left(\frac{(u_p - c_{ip})^2}{\sigma_{ip}^2}\right)\right). \qquad (9)$$

   Then, the input u_r is reconsidered in the final membership function and the following expression is obtained,.

$$\mu_i(\underline{u}) = exp\left(-\frac{1}{2}\left(\frac{(u_r - c_{ir})^2}{\sigma_{ir}^2}\right)\right) + \mu_c. \qquad (10)$$

3. *Create the linear membership function for the required input as from equation (8).*

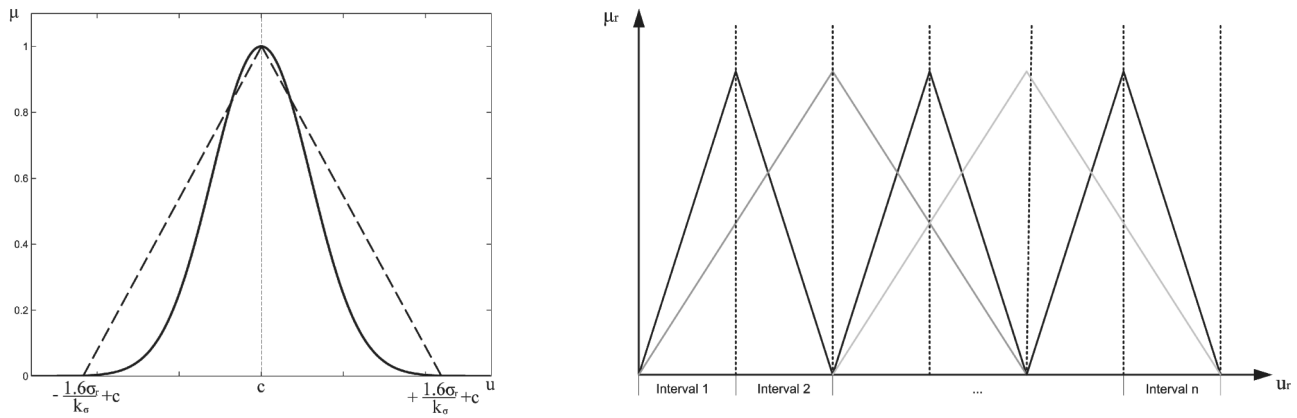

Fig. 4. Left: Linear and non-linear validity function. Right: Linear functions on the intervals.

4. *Partition the input space u_r.*
The input space of u_r is partitioned into q search intervals, which are used in the later estimation of u_r. Every interval describes the validity of half of the local linear model. Thus, the input space of every LLM is divided into two intervals. This is necessary due to the structure of the new linear membership functions, because they consist of two linear functions as mentioned above. For every interval, a "left function" and a "right function" are considered (Fig. 7). For the sake of brevity, equation (8) is represented as the following:

$$\mu(u_r)_{i,r} = \begin{cases} \mu(u_r)_{i,r,1} \\ \mu(u_r)_{i,r,2}. \end{cases} \quad (11)$$

5. *In the following loop, consider every interval for a possible solution of u_r.*

- Determine which of i membership functions $\mu(u_r)_{i,r}$ are valid for the currently considered interval, by checking every spline. $\mu(u_r)_{i,r,1}$ is valid if

  $$\mu(interval\_left)_{i,r,1} \geq 0 \wedge \mu(interval\_right)_{i,r,1} \geq 1$$

  $\mu(u_r)_{i,r,2}$ is valid if

  $$\mu(interval\_left)_{i,r,2} \geq 0 \wedge \mu(interval\_right)_{i,r,2} \geq 0,$$

  where $\wedge$ indicates the "and" logical function.

- Use the valid membership spline functions to create validity functions for each local linear model. Taking the previously calculated part of the membership function i, $\mu_{i,calc}$ as in (10), and sum it with the valid linear spline membership function i $\mu(u_r)_{i,r,\{1,2\}}$, where $\mu(u_r)_{i,r,\{1,2\}}$ represents a valid spline membership function within the range of functions. This yields:

  $$\mu(u_r)_i = \mu(u_r)_{i,r,\{1,2\}} + \mu_{i,calc}$$

  and $$\Phi(u_r)_i = \frac{(u_r)_i}{\sum_{j=1}^{M} \mu(u_r)_j}.$$

  If there are no valid linear membership functions for a local linear model, then the model will not be considered for further actions.

- Initially create the output function for every local linear model by multiplying its validity function with the local linear model function:

  $$\hat{y}(u_r)_{LLM,i} = \hat{y}(u_r)_i = \Phi(u_r)_i.$$

  Next, sum the output functions to create the model output:

  $$\hat{y}(u_r) = \sum_{i=1}^{M} \hat{y}(u_r)_{LLM,i}.$$

  Finally, equate the model output function to the desired model output value, and solve the resulting equation with respect to the variable u_r:

  $$y = \hat{y}(u_r).$$

- Verify that the calculated u_r is inside the currently considered interval.

  $$interval\_left \leq u_r \leq interval\_right.$$

  If so, accept it as one possible solution. If not, disregard it.

Due to the structure of the validity functions, an inversion of the model is possible only within the input space of the required variable. Beyond these borders, the model input will drift to zero, which is comparable to the normal LOLIMOT behaviour. That is, once a *work nominal point* is chosen, the inversion is possible within the input domain. The worst case is if the working nominal point is close to the border of the domain. If so, a more suitable division of the input space is needed, as is described in the next section.

2.3. Boosting

With a continuous system, it is possible to accelerate the algorithm's runtime behaviour by reducing the time needed to select the validity functions during the inversion. This makes it necessary to perform some off-line calculations on the linear validity functions, which otherwise remain unchanged during the execution of the inversion. For every linear validity function, two points are calculated and saved in a look-up table: the starting point p_s and the point p_e where the function intersects the input domain axis. This information is used at runtime to pre-select the relevant validity functions; therefore, a region of interest (ROI) around the previously calculated desired input value $u_r(k-1)$ has to be defined as an interval, e.g.,

$$ROI(u_r(k-1)) = [u_r(k-1) - c \cdot u_r(k-1) \quad u_r(k-1) + c \cdot u_r(k-1)],$$

where c is a parameter describing the size of the region. If the function crosses, starts, or ends in the ROI, the validity function should be considered.

A logic validity function (LVF) can be defined as follows:

$$\begin{aligned} LVF = {} & (p_s \leq ROI_{max} \vee p_s \geq ROI_{min}) \vee \\ & (p_e \leq ROI_{max} \vee p_e \geq ROI_{min}) \vee \\ & (p_s > ROI_{max} \wedge p_e < ROI_{min}) \vee \\ & (p_e > ROI_{max} \wedge p_s < ROI_{min}), \end{aligned} \quad (12)$$

where "$\vee$" and "$\wedge$" indicate the "or" and the "and" logical functions, respectively. If the variable LVF assumes a value equal to 1, then the validity function should be considered; if the variable LVF assumes a value equal to 0, then the validity function should be not considered. The proposed boosting algorithm is similar to the algorithms used to solve clipping problems in computer graphics, in which the ROI states the camera field of view.

3. Analysis of the algorithm

3.1. Stability

The algorithm is mostly stable; however, due to the limited character of the linear validity functions, it is possible that no solution will be found in the border regions of the model.

In that case, there are two possible solutions:

- The first solution is to use a more complex approximation of the validity function, e.g., a linear spline consisting of n line segments. This will increase the computational effort. In fact, each line segment represents a single interval, which must be considered to solve the inversion problem. Since the validity function is symmetric, the effort is increased by a factor of two. Applying the proposed boosting method will reduce the complexity.

- Another solution is to adjust the zeros of the linear approximation function so that the function will cover a larger region of the input domain. This will reduce the accuracy of the estimated input value, but it keeps the computational costs low.

3.2. Ambiguities

Since the algorithm solves a squared equation, it is possible to obtain two solutions for each of the q intervals if the solutions are valid. In the worst case this results in $2 * q$ solutions. Therefore, we introduce a decision criterion to select the correct input value.

If the system is continuous, as it is in most real applications, this decision criterion could be based on the previous input value $u_r(k-1)$. Other criteria could also be used.

3.3. Accuracy

The accuracy of the inversion is mainly influenced by the linear validity function. Therefore, a linear/linear spline validity function will lead to more accurate results compared with the simplest linear function. As mentioned above, this will increase the computational complexity, so the best compromise between speed and accuracy has to be found.

Boosting can be used to reduce these negative effects. By using the simplest validity functions, with one linear spline per side, a difference of 10-15 percent between the predicted input value and the real input value can occur in the worst case. However, the result can be improved by applying a numerical optimization process.

4. A real application: The Throttle Valve

The training data was obtained using measurements obtained from an experimental setup.

We examined the manifold of the throttle angle using a network consisting of three neurons. We set up the inversion problem to depend on four inputs (Fig. 5): the input voltage (u_1), the ambient air pressure in mbar (u_2), the manifold air-mass pressure in mbar (u_3), and the ambient temperature in Celsius (u_4). Normally the model output would be the manifold air mass flow in kg/h, but here we used the throttle angle as output to correspond with the model presented in equations (1), (2) and (3). The model is trained with a k sigma equal to 0.33. In this example, the inversion is tested by the throttle angle, depending on the desired angular trajectory (Fig. 5). The scheme of Fig. 5 shows two possible simulations corresponding to the control schemes represented in Fig. 6 and 7. Fig. 8 shows a phase of acceleration and deceleration using a control scheme with the inversion defined in equation (4), and using a proportional integral derivative (PID) control in feedback configuration. Fig. 9 shows a simulation using a control scheme with an Inverse model from the proposed neural network, and using a PID control in feedback configuration. Both cases are tested with the same PID controller parameters. The input parameters for u_2, u_3 and u_4 are chosen as static values.

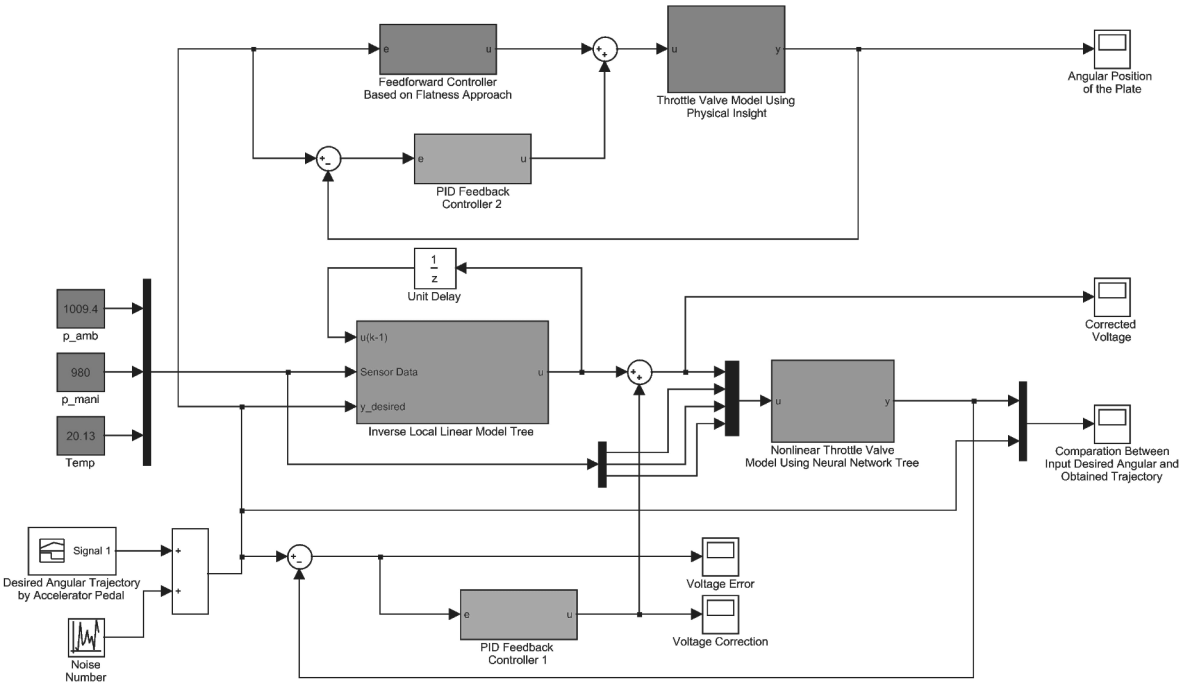

Fig. 5. Simulink-block scheme of the used model.

5. Conclusions and Outlook

This paper describes a powerful algorithm for the inversion of local linear model trees at runtime, and we apply the method to an automotive throttle valve control problem as an attractive example. The algorithm was analysed in terms of stability and accuracy. The results demonstrated that the inverse local linear model tree based upon Takagi-Sugeno models can be integrated into a control loop, and we obtained very good noise reduction performances. To extend this work, possible future efforts should focus on error detection and diagnostics, especially model-based system diagnostics.

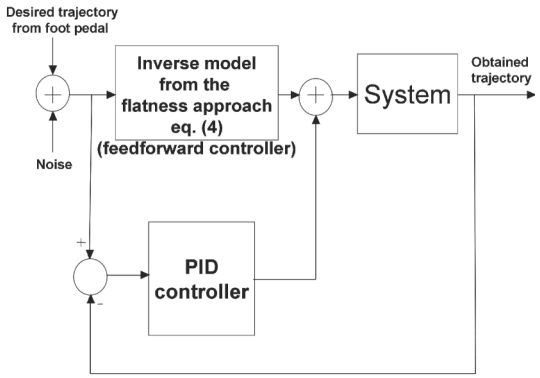

Fig. 6. Control scheme with inversion defined in (4) and using a PID control in feedback configuration.

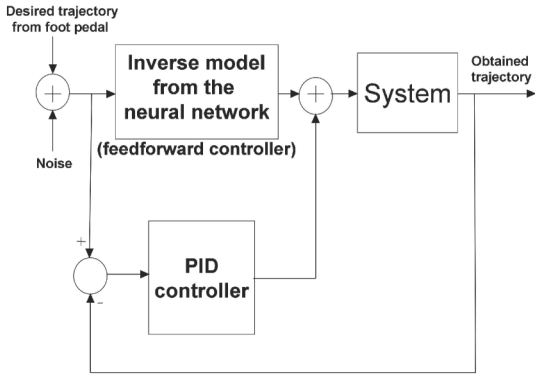

Fig. 7. Control scheme with inverse model from the proposed neural network and using a PID control in feedback configuration.

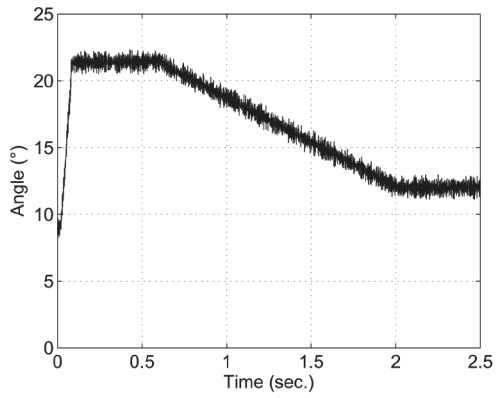

Fig. 8. Desired and obtained angular position with noise in superposition using the control scheme as in Fig. 6.

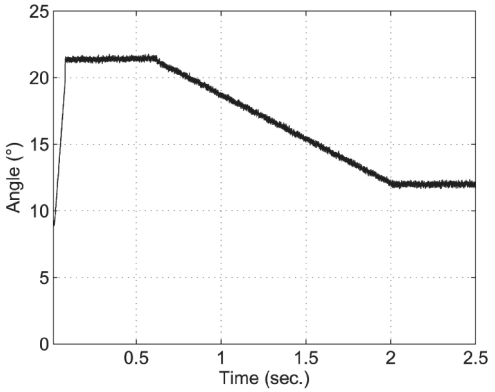

Fig. 9. Desired and obtained angular position with noise in superposition using the control scheme as in Fig. 7.

AUTHORS

Mirko Nentwig* and Paolo Mercorelli - University of Applied Sciences, Braunschweig/Wolfenbuettel, Wolfsburg, Germany. E-mails: mail@mnentwig.de; p.mercorelli@fh-wolfenbuettel.de.

* Corresponding author

References

[1] Nakano K., *et. al.*, "Modelling and observer-based sliding-mode control of electronic throttle systems", *ECTI Trans. Electrical Eng., Electronics and Communications*, vol. 4, no. 1, 2006, pp. 22-28.

[2] Fink A., Nelles O., "Nonlinear internal model control based on local linear neural networks". In: *IEEE Systems, Man, and Cybernetics*, Tucson, USA, 2001.

[3] Fink A., Nelles O., Fischer M., "Linearization based and local model based controller design". In: *European Control Conference (ECC)*, Karlsruhe, Germany, 1999.

[4] Fink A., Toepfer A., *On the inversion of nonlinear models. Technical report*, University of Darmstadt, 2003.

[5] Fink A., Toepfer S., Isermann R., "Neuro and neuro-fuzzy identification for model-based control". In: *IFAC Workshop on Advanced Fuzzy/Neural Control*, Valencia, Spain, vol. Q, 2001, pp. 111-116.

[6] Fink A., Toepfer S., Isermann R., "Nonlinear model-based control with local linear neuro-fuzzy models", *Archive of Applied Mechanics*, vol. 72, no. 11-12, 2003, pp. 911-922.

[7] Fischer M., Nelles O., Fink A., "Supervision of non-linear adaptive controllers based on fuzzy models". In: *14th IFAC World Congress*, Beijing, China, vol. Q, 1999, pp. 335-340.

[8] Mercorelli P., "An optimal minimum phase approximating pd regulator for robust control of a throttle plate". In: *45th IEEE Conference on Decision and Control (CDC2006)*, San Diego (USA), 13th-15th December, 2006.

[9] Nelles O., *Nonlinear System Identification with Local Linear Neuro-Fuzzy Models*. Shaker Verlag, 1999.

[10] Nelles O., *Nonlinear System Identification*. Springer Verlag, 2001.

[11] Nelles O., Fink A., Isermann R. "Local linear model trees (lolimot) toolbox for nonlinear system identification". In: *12th IFAC Symposium on System Identification (SYSID)*, Santa Barbara, USA, 2000.

[12] Nentwig M., Mercorelli P., "A matlab/simulink tool-box for inversion of local linear - model trees". *in press*.

[13] Rossi C., Tilli A., Tonielli A., "Robust control of a throttle body for drive by wire operation of automotive engines". In: *IEEE Trans. Contr. Syst. Technology*, vol. 8, no. 6, 2000, pp. 993-1002.