# Method to design a neural network with minimal number of neurons for approximation problems

**H. Ruppert*, A. Krug*, Y. A. W. Shardt****

** Faculty of Electrical Engineering, Heilbronn University of Applied Sciences*
*Heilbronn, Germany, (e-mail: {holger.ruppert, andreas.krug}@hs-heilbronn.de)*

*** Faculty of Computer Science and Automation, TU Ilmenau*
*Ilmenau, Germany, (e-mail: yuri.shardt@tu-ilmenau.de)*

**Abstract**: — The widespread use of neural networks to model complex processes requires that a parsimonious model of the process be obtained. One of the main variables in neural networks is the number of neurons in the hidden layer. Selecting an inappropriate number of neurons can lead to over- or underfitting. Therefore, a method is required which determines the appropriate number of neurons in order to approximate a defined system response or time function. This paper presents a proposition to determine the appropriate number of neurons in a feedforward neural network, based on the number of inflection points included in the system response or the time function. The results show that the proposed method has marginal approximation errors (no underfitting) and overfitting can never occur because the minimal number of neurons for the approximation problem is used. To verify the effectiveness of this method, simulations were carried out on a second-order system with and without noise, the Lotka-Volterra equations, and the Runge function.

*Keywords*: inflection point, number of neurons, feed forward neural network, approximation, hyperbolic tangents

## 1. INTRODUCTION

Neural networks have a wide range of applications, for example in controlling dynamic systems like electrical motors, description and simulation of highly complex dynamic systems (Hunter *et al*., 2012), approximation and prediction of time series (Mhaskar *et al*., 1995), regression (Gronhold *et al*. 2005 and Bishop, 1995) and sensor applications (Kadlec, *et al*., 2009). In this paper, we manly deal with approximation behavior of neural networks for sensor applications.

The primary purpose of the sensors is to deliver data for process monitoring and control. However, some process variables cannot easily be recorded with hardware sensors because the process variable is not available (Xu *et al*., 2018) or the hardware sensors are disproportionately expensive (Fortuna *et al*., 2006). Thus, researchers have to use the data measured by hardware sensors to build predictive models. In the context of the process industry, these predictive models are called soft sensors (Kadlec *et al*., 2009).

At a very general level, one can differentiate between model- and data-based soft sensors. The model-based soft sensor is most commonly based on physical equations describing the physical principles underlying the process, also called white-box models because they have full phenomenological knowledge about the process background (Shardt, 2015). On the other hand, purely data-driven models are called black-box models, because the model itself has no knowledge about the process and is based on empirical observations of the process. One of the most popular methods for modelling data-driven soft sensors is using a regression model based on an artificial neural network (Kadlec *et al*., 2009).

However, there are two major problems in the application of neural networks to modelling processes (Mhaskar *et al*., 1995). First, it is necessary to determine the number of neurons required to achieve the approximation of the target function within a given margin of tolerance. Secondly, it is necessary to develop algorithms to actually construct the approximating networks.

Therefore, many papers deal with how the architecture of a neural network has to be chosen in order to be able to achieve acceptable results. Neural networks with no hidden layers lack the capability to approximate nonlinear functions (Scarselli *et al*., 1998). Cybenko (1989) proved, that networks with one hidden layer and the sigmoid activation function could be universal approximators. Hornik *et al*. (1989) extended Cybenko's work to feedforward neural networks (FNN) with other activation functions. The universal approximation property of an FNN has been studied using tools from functional analysis *(e.g.*, the Hahn-Banach theorem in Rudin (1973)) and real analysis (*e.g.*, the Sprecher-Kolmogorov theorem in Sprecher (1965) and Kolmogorov (1957)). In Lin et al. (2021), the relationship between a multilayer perceptron regressor and the piecewise polynomial approximator was studied, and based on this relationship, a multilayer perceptron construction method was proposed. The Radon transform (Carroll, 1989), and the Fourier distribution and series (Barron, 1993) have been used for the construction of a FNN.

Thus, this paper proposes and examines a new method by which the minimal number of neurons in a FNN can be specified based on the number of inflection points in the system response or time function. Furthermore, the proposed approach is tested on simulated second-order system with and without noise, the Lotka-Volterra equations, and the Runge function.

## 2. MATHEMATICAL DESCRIPTION OF THE FEEDFORWARD NEURAL NETWORK

Let us consider the propagation through a neuron as shown in Fig. 1, where time $t$, weight $w$, and bias $b$ are inputs. With these inputs the propagation function $u$ can be described by

$$u = wt + b. \tag{1}$$

The activation function $a(u)$ is equivalent to the output $\tilde{f}$ of the neuron.

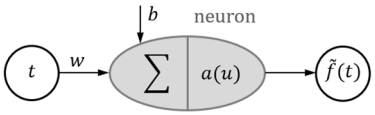


Figure 1: Propagation through one neuron in a FNN

The nonlinear hyperbolic tangent function, commonly used to approximate nonlinear functions (Aggarwal, 2018), given as,

$$a(u) = \frac{2}{1 + e^{-2u}} + 1 \tag{2}$$

is chosen here as the activation function. The propagation function $u$ works as a scaling factor (Silva *et al*., 2017) that horizontally stretches or shrinks $a(u)$ with the weight $w$. The bias $b$ performs a horizontal translation. If the sign of $w$ changes, $a(u)$ is reflected over the *y*-axis.

As can see from the consideration above, a FNN with one neuron can only describe a time behavior with the shape of the activation function. With a FNN as shown in Fig. 2, with one hidden layer {1} with *K* neurons, and the output layer {2} with one neuron with multiple inputs, other time behaviors with higher complexity can also be described by superposing.

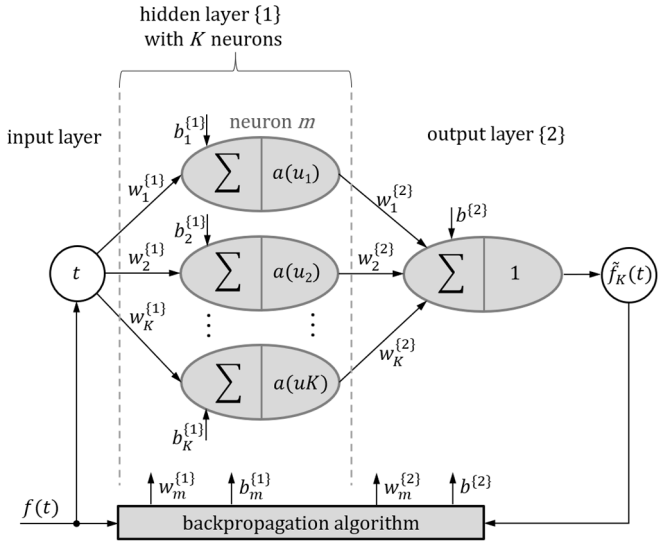


Figure 2: Arrangement of neurons, weights, biases and activation functions to describe the FNN

The $m$ neurons are part of the hidden layer and described by their individual weights $w_m^{\{1\}}$, biases $b_m^{\{1\}}$ and propagation functions $u_m$. Therefore, (1) becomes

$$u_m = w_m^{\{1\}} t + b_m^{\{1\}}. \tag{3}$$

The activation function for each neuron $m$ is given by $a(u_m)$. The approximation $\tilde{f}_K$ of the nonlinear time functions by a FNN with the input $f$, with $K$ neurons in the hidden layer and one neuron in the output layer, including a linear activation function gives

$$\tilde{f}_K = \left( \sum_{m=1}^{K} w_m^{\{2\}} a(u_m) \right) + b^{\{2\}}, \tag{4}$$

where $w_m^{\{2\}}$ is the individual weight of each neuron of the second layer and $b^{\{2\}}$ is the total bias of the second layer. Both can be seen as a vertical scaling factor. The complete structure of a FNN as shown in Fig. 2 and used in this paper, describes (4). The approximated function $\tilde{f}_K$ is a set of $K$ superpose hyperbolic tangent functions:

$$\begin{aligned} \tilde{f}_K = &\left( \frac{2}{1 + e^{-2u_1}} + 1 \right) w_1^{\{2\}} + \left( \frac{2}{1 + e^{-2u_2}} + 1 \right) w_2^{\{2\}} + \\ &\ldots + \left( \frac{2}{1 + e^{-2u_K}} + 1 \right) w_K^{\{2\}} + b^{\{2\}}. \end{aligned} \tag{5}$$

To adjust the weights $w_m^{\{1\}}$ and $w_m^{\{2\}}$, the backpropagation algorithm is commonly used (Hecht, 1989). Using $J$ optimization steps, the weights $w_m^{\{1\}}$ and $w_m^{\{2\}}$ are optimized by looping through the following equations:

$$\begin{aligned} \frac{\partial e_i^{\{2\}}}{\partial w_{m,i}^{\{2\}}} &= \frac{\partial \frac{1}{2}\left(f - \tilde{f}_i\right)^2}{\partial \tilde{f}_i} \frac{\partial \tilde{f}_i}{\partial w_{m,i}^{\{2\}}} \\ \frac{\partial e_i^{\{1\}}}{\partial w_{m,i}^{\{1\}}} &= \frac{\partial \frac{1}{2}\left(f - \tilde{f}_i\right)^2}{\partial \tilde{f}_i} \frac{\partial \tilde{f}_i}{\partial a(u_{m,i}) w_{m,i}^{\{2\}}} \frac{\partial a(u_{m,i}) w_{m,i}^{\{2\}}}{\partial w_{m,i}^{\{1\}}}. \end{aligned} \tag{6}$$

In (6) and (7), the functions $e_i^{\{1\}}$ and $e_i^{\{2\}}$ are the error functions for each optimization step $i$. With the defined gradients for each parameter, the weights $w_{m,i}^{\{1\}}$, $w_{m,i}^{\{2\}}$ are optimized and updated at each step $i$ using the gradient descent method (Benvenuto *et al*., (1992), that is,

$$\begin{aligned} w_{m,i}^{\{1\}} &= w_{m,i-1}^{\{1\}} - \eta_i \frac{\partial e_i^{\{1\}}}{\partial w_{m,i}^{\{1\}}} \\ w_{m,i}^{\{2\}} &= w_{m,i-1}^{\{2\}} - \eta_i \frac{\partial e_i^{\{2\}}}{\partial w_{m,i}^{\{2\}}}, \end{aligned} \tag{7}$$

where $w_{m,i}^{\{1\}}$ and $w_{m,i}^{\{2\}}$ are the weights of the current optimization step $i$ for respectively the hidden and the output layer, and $w_{m,i-1}^{\{1\}}$, $w_{m,i-1}^{\{2\}}$ are the weights of the last optimization step. The variable learning rate for each

optimization step $i$ is given by $\eta_i$. If no further improvement of the approximation can be brought about by the algorithm, $w_{m,i}^{\{1\}}$ converges to the weight $w_m^{\{1\}}$ and $w_{m,i}^{\{2\}}$ converges to the weight $w_m^{\{2\}}$ and the termination criterion has been reached. To adjust the biases $b_m^{\{1\}}$ and $b^{\{2\}}$, the same procedure is used.

## 3. APPROXIMATION OF A SECOND-ORDER SYSTEM BY A FEEDFORWARD NEURAL NETWORK

Let us consider the time series resulting from the simulation of a second-order system of the form

$$y + 2DT\frac{\mathrm{d}y}{\mathrm{d}t} + T^2\frac{\mathrm{d}^2y}{\mathrm{d}t^2} = Px, \qquad (8)$$

where the damping constant $D = 0.268$, the time constant $T = 0.193$, and the gain $P = 1$. The second-order system is a common model for describing damped oscillation, *e.g.* a mass-spring-damper systems and RLC circuits. The process will be excited using a step of magnitude 1 for the variable $x$. The output variable $y$ describes then the step response of the process.

The approximations $\tilde{y}_K$ of the step response $y$ were generated with the FNN described above. The results of modelling the process using $K = 1$, 2, and 3 neurons are shown in Fig. 3.

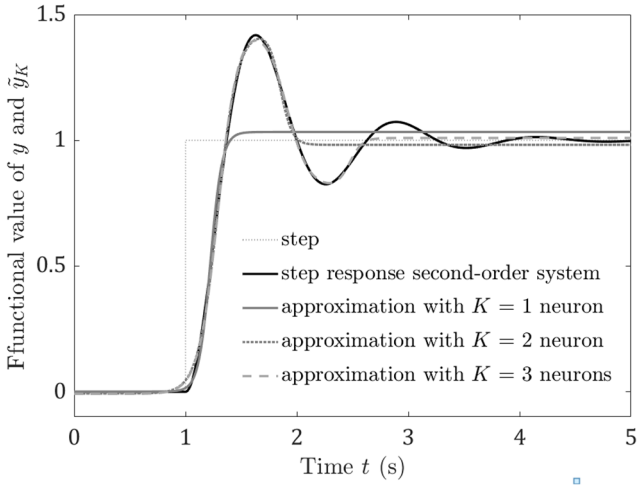


Figure 3: Step response of a second-order system with FNN approximations $\tilde{y}_1$, $\tilde{y}_2$, and $\tilde{y}_3$ for different $K$

As is shown by (5), the approximation $\tilde{y}_1$ resembles a hyperbolic tangent, which has a form similar to that of the integral symbol ($\int$). The backpropagation algorithm adjusts $\tilde{y}_1$, that is, the hyperbolic tangent, in such a way that when all the deviations between $y$ and $\tilde{y}_1$ are added together, the smallest sum results. Therefore, $\tilde{y}_1$ approaches the first positive edge of $y$ until this tends towards a final value that is roughly the mean value of the over- and undershoots.

A better approximation can be achieved by superimposing additional hyperbolic tangent functions, as described in (5). The first overshoot can be approximated by $\tilde{y}_2$ with $K = 2$ neurons, and the subsequent undershoot can also be approximated by $\tilde{y}_3$ with a third neuron. It is therefore obvious that for every further inflection point in the step response $y$, another neuron is necessary, in order to guarantee a complete approximation over the considered interval.

Let us consider the step response between two extrema, defined by a peak in the overshoot and a peak in the subsequent undershoot, then it is noticeable that this area has the shape of an integral symbol or its reflection about the $y$-axis. Since each neuron only forms one hyperbolic tangent of the form of an integral symbol, the number of neurons must be equal to the number of inflection points $W$. In this case, $W = 5$ inflection points are present. Thus, the result of the approximation with a FNN with $K = 5$ neurons is shown in Fig. 4.

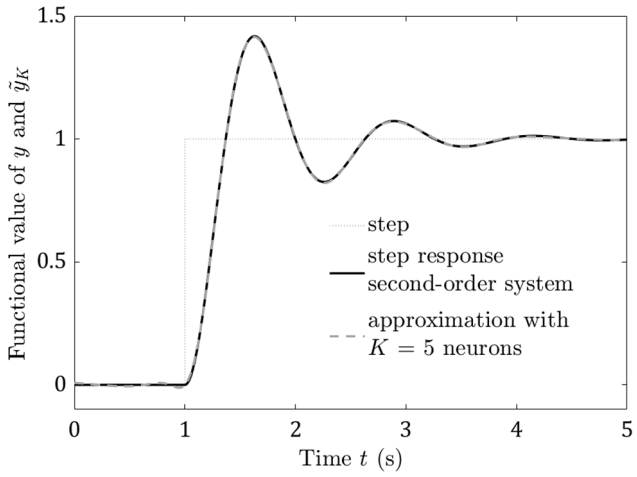


Figure 4: Step response of a second-order system with FNN approximation $\tilde{y}_5$ for $K = 5$

A sufficiently good approximation is achieved with $K = 5$ neurons (see Fig. 4), where $R_5 = 0.999$ and the deviation between $y$ and its approximation $\tilde{y}_5$ is smaller than 1.5%. The coefficient of regression $R_K$ measures the correlation between $y$ and $\tilde{y}_K$, where $R = 1$ denotes a strong relationship between $y$ and $\tilde{y}_K$, and $R = 0$ no relationship. The approximation error $\Delta_K = (y - \tilde{y}_K)$ is shown in Fig. 5.

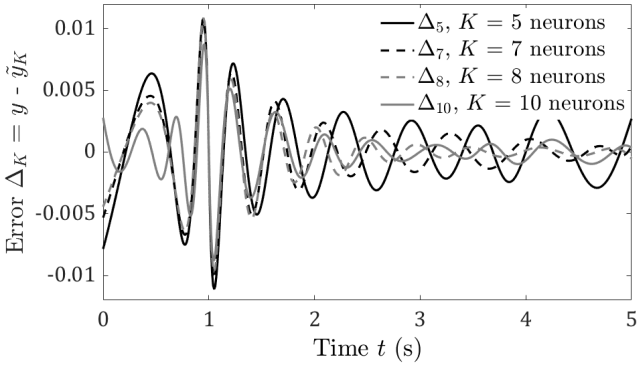


Figure 5: Approximation error $\Delta_K$ produced by the FNN with different $K$.

Every approximation error $\Delta_K$ oscillates between $\pm 1.2\%$. The quality of the approximation improves only marginally with a further increase in the number of neurons, since the coefficient of regression remains at $R_K = 0.999$ for $K = 5$, 7, 8, and 10. Furthermore, the approximation error (see Fig. 5) remains roughly the same despite an increasing number of neurons.

## 4. PROPOSITION REGARDING THE APPROXIMATION OF OSCILLATING FUNCTIONS

Based on the experiment above, let us consider a proposition that defines the minimum number of neurons required in a FNN. Let us divide a time function *f* in the interval $[L, U]$ into $M$ subintervals $[\Delta L_m, \Delta U_m] \in [L, U]$ with the index $m \in M$. If every interval boundary is determined by $f'(\Delta L_m) = 0$ and $f'(\Delta U_m) = 0$, where $f' = \mathrm{d}f/\mathrm{d}t$, $f$ only monotonically increases ($\mathrm{d}f/\mathrm{d}t \geq 0$ at each point in $[\Delta L_m, \Delta U_m]$) or monotonically decreasing ($\mathrm{d}f/\mathrm{d}t < 0$ at each point in $[\Delta L_m, \Delta U_m]$ ). Then, $f$ has exactly one inflection point in the subinterval $[\Delta L_m, \Delta U_m]$ and $f$ has the form of an integral symbol or its reflection about the *y*-axes in every subinterval.

If the activation function has the form of the integral symbol or its reflection about the *y*-axis, this is strictly monotonically increasing or decreasing like the hyperbolic tangent function. This can only approximate one interval $[\Delta L_m, \Delta U_m]$ in *f*, in which these is also a strict monotonic increase or decrease. This means that the hidden layer of a FNN must have as many neurons $K$ as there are inflection points $W$ in *f*, since a neuron can only generate a single, monotonically increasing or decreasing function.

The number of inflection points $W$ in the function $f$ is given by the number of subintervals, *i.e.*

$$K = W = \sum_{m=1}^{M} 1^m. \tag{9}$$

If the number of neurons corresponds to the number of inflection points in *f*, in the interval $[L, U]$ we are considering, the slope and the position of the inflection point of the function $a(u_m)$ can be adjusted with $u_m$ for each inflection point in *f*. The minimal number of neurons that have to be contained in the hidden layer in a FNN, in order to be able to approximate $f$, is described by $K$.

## 5. VERIFICATION OF THE PROPOSITION

The proposition verified using the Lotka-Volterra equations, the Runge function, and a noisy step response of a second-order system. The Lotka-Volterra equations describe the interaction between predator and prey populations, consisting of two nonlinear, coupled first-order differential equations, that is,

$$\frac{\mathrm{d}n}{\mathrm{d}t} = n\left[C\left(1 - \frac{n}{G}\right) - Dp\right]$$

$$\frac{\mathrm{d}p}{\mathrm{d}t} = p(Bn - A), \tag{10}$$

where *n* is the number of prey, *p* the number of predators, and $A = 5$, $B = 0.05$, $C = 8$, $D = 0.3$ and $G = 300$ are user-defined parameters.

Based on the proposition, both FNNs for the approximation of $n$ and $p$ are implemented with $K = 5$ neurons, because $W = 5$ inflection points can be seen in the simulation of the Lotka-Volterra equations, for the give parameters. A change in the parameters leads to a different behavior. In Fig. 6, $n$, $p$ and their approximations $\tilde{n}$, $\tilde{p}$ are shown.

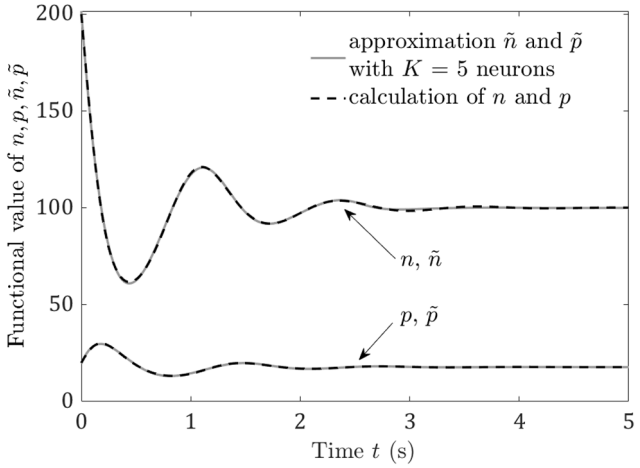


Figure 6: Results of the simulation of $n$ and $p$ and their approximation $\tilde{n}, \tilde{p}$

Both approximations have small deviations with a maximum error $|\Delta_{max}| = 0.2\%$. The proposed method to choose the number $K$ of neurons with the number $W$ of inflection points also applies to nonlinear systems like Lotka-Volterra.

The Runge function is very difficult to approximate with a polynomial function because of Runge's phenomenon, where higher-order polynomials oscillate at the edges of an interval if the step size between the interpolation points is constant. Therefore, let us consider how well a FNN can approximate this function. The Runge function is given by

$$f = \frac{1}{1 + t^2} \tag{11}$$

Based on the proposition, one FNN for the approximation $\tilde{f}_K$ of the Runge function $f$ is implemented with $K = 2$ neurons, because $W = 2$ inflection points can be seen in the simulation of the Runge function and another FNN implemented with $K = 8$ neurons. The results are shown in Fig. 7.

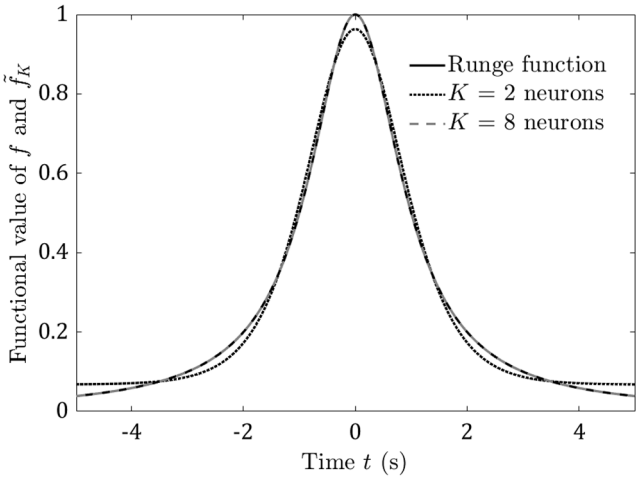


Figure 7: Results arising from approximating the Runge function with two and eight neurons

The approximation $\tilde{f}_2$ with $K = 2$ neurons has a smaller amplitude than the Runge function and for $t > 1.5$ s a different

shape occurs. Between the approximation $\tilde{f}_8$ and the Runge function, no different are visible. In Fig. 8 the approximation error $\Delta = f - \tilde{f}_K$ is shown.

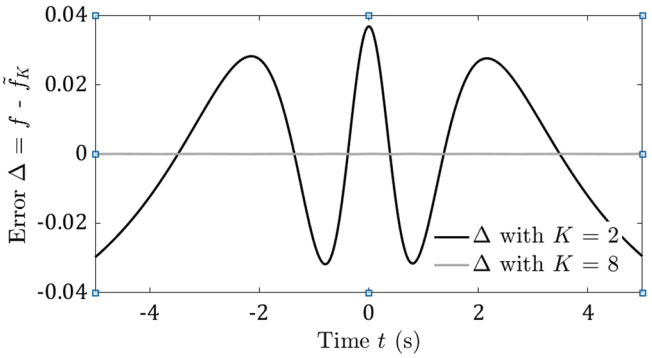


Figure 8: Approximation errors produced by the FNN with a different number $K$ of neurons in the hidden layer

Since the approximation error between the Runge function and the approximation $\tilde{f}_8$ is almost zero, the statement is confirmed that no differences can be seen between both functions. The approximation error between the approximation $\tilde{f}_2$ and the Runge function oscillates, because the Runge function intersects the approximation. The approximation error is minimized using the FNN with the higher number of neurons.

However, it cannot simply be assumed, that by increasing the number of neurons, the quality of the approximation increases. This becomes particularly problematic, when dealing with data corrupted by noise.

To simulate these, noise is added to $y$ in Equation (8). In Simulink, the band-limited white noise block is used for this with a noise power $P_n = 0.00015$ and a sample time $T_s = 0.01$ s. This generates normally distributed random numbers around the step response $y$. The simulation results of the noisy step response $y_{noisy}$ and their approximations $\tilde{y}_{5,noisy}$ and $\tilde{y}_{100,noisy}$ are shown in Fig. 9 and 10.

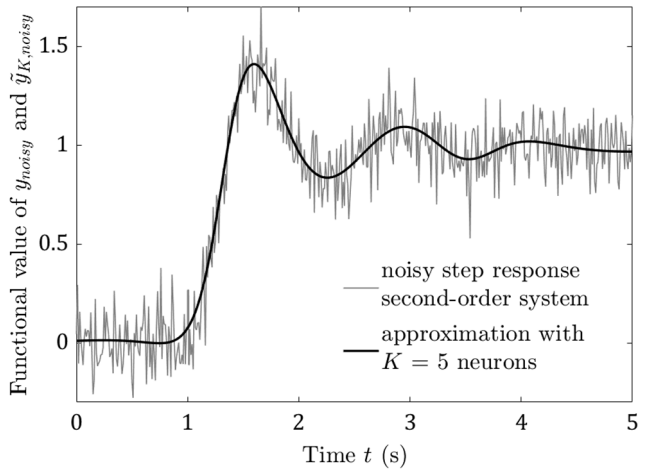


Figure 9: Simulation results of noisy step response $y_{noisy}$ of the second-order system with its approximation $\tilde{y}_{5,noisy}$

The approximation $\tilde{y}_{5,noisy}$ corresponds to a smoothed signal that has the characteristic of the unnoisy step response of the second-order system $y$.

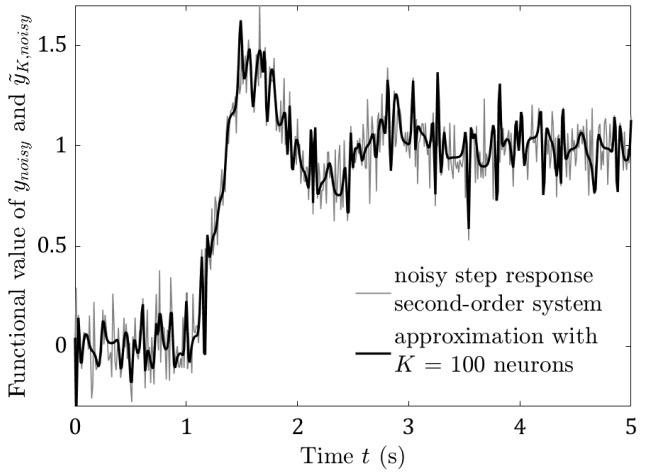


Figure 10: Simulation results of noisy step response $y_{noisy}$ of the second-order system with its approximation $\tilde{y}_{100}$

The approximation $\tilde{y}_{100,noisy}$ also contains higher frequency components as can be seen in Fig. 10, so that it can be assumed that the neural network becomes more sensitive to higher frequencies by increasing the number of neurons. Conversely, this means that this method to choose the number of neurons also suppresses unwanted noise, when approximating a system response. If the FNN described here, is designed with the minimum number of required neurons, it can also act as a filter. It is easy to see that the approximation $\tilde{y}_{5,noisy}$ is closer to the true step response (see Fig. 4) than the approximation $\tilde{y}_{100,noisy}$. Fig. 11 shows $y$ and the approximation $\tilde{y}_{5,noisy}$.

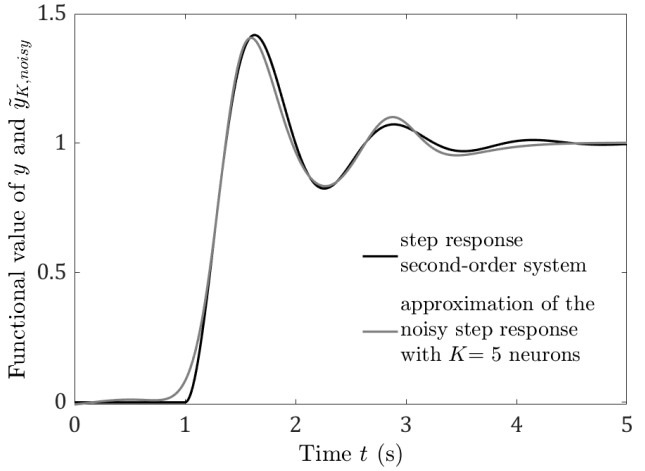


Figure 11: Comparison between the approximation of the noisy step response $\tilde{y}_{5,noisy}$ and the step response $y$

If we compare the approximations $\tilde{y}_{5,noisy}$ and $\tilde{y}_5$ with the step response $y$ (see Fig. 11 and Fig. 4), it can be retained that both approximations $\tilde{y}_{5,noisy}$, and $\tilde{y}_5$, which the FNN delivers, have the same characteristics, despite different input signals. Thus, an FFN designed with this method can find the true function in a noisy signal if the characteristics of a system response are known. In practice, this can be a helpful tool for system identification, since measured values are usually noisy.

## 6. CONCLUSIONS

In this paper, a new method for designing a neural network for approximating a data set was presented. This method uses inflection point analysis to determine the number of neurons in a feedforward neural network with one hidden layer and a hyperbolic tangent activation. The advantage of the method is that no regularization is required since the smallest number of neurons is used automatically.

Simulations have shown that the method can define the minimal number of neurons, and hence, approximate linear and nonlinear functions with small error. It can be used for a large group of linear and nonlinear systems, with the assumption that the function has derivatives on the domain, where the approximation is desired. In summary, by using the hyperbolic tangent as activation function, a FNN with one neuron in the hidden layer can only approximate the time behavior of nonoscillating system responses, which are typically the solutions of linear and nonlinear first-order differential equations. To approximate linear or nonlinear differential equations of higher order, especially with oscillating behavior, two or more neurons are needed. It should be noted that other types of activation functions will produce different results.

Future work will clarify, whether this method can be only used for a special class of functions, especially for differential equations, which have the exponential function as a solution or whether this method can also be used for any nonlinear differential equations. In addition, it must be verified, whether an automated creation of a neural network is possible on the basis of this method, which then works as a regression model in a soft sensor. Moreover, it will be checked in further works, whether the proposed method is applicable to deep neural networks with more than one hidden layer.

## REFERENCES


Aggarwal, C.C. (2018). *Neural Networks and Deep Learning.* Springer International Publishing.

Barron A.R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inf. theory*, 39(3), 930-945.

Benvenuto N. and F. Piazza (1992). On the complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 40(4), 967-969.

Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Carroll S. (1989). Construction of neural networks using the radon transform. *Proc. IEEE Int. Conf. Neural Network*, 1, 607-611.

Cybenko G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Control, Signals Syst*, 2(4), 303-314.

Fortuna L., Graziani S., Rizzo A., Xibilia M.G. (2006). *Soft Sensor for Monitoring and Control of Industrial Processes*. Springer-Verlag, London.

Gronholdt, L. and Martensen, A. (2005). Analysing customer satisfaction data: acomparison of regression and artificial neural networks. *International Journal of Market Research*, 47(2), 121-130.

Hecht-Nielsen R. (1989). Theory of the backpropagation neural network. *International 1989 Joint Conference on Neural Networks*, 1, 593-605.

Hornik K., Stinchcombe M., and White H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw*., 2(5), 359-366.

Hunter D., Yu H., Pukish M.S, Kolbusz J. and Wilamowski B.M. (2012). Selection of Proper Neural Network Sizes and Architectures - A Comparative Study. *IEEE Transactions on Industrial Informatics*, 8(2), 228-240.

Kadlec, P., Gabrys, B. and Strandt, S. (2009). Data-driven Soft Sensors in the Process Industry. *Computers and Chemical Engineering*, 33(4), 795-814.

Kolmogorov A.N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk, Russian Acad. Sci*., 114(5), 953-956.

Lin R., You S., Rao R. and Kuo C.-C. J. (2021). On Relationship of Multilayer Perceptrons and Piecewise Polynomial Approximators. *IEEE Signal Processing Letters*, 28, 1813-1817.

Mhaskar H.N. and Khachikyan L. (1995). Neural networks for function approximation. *Proceedings of 1995 IEEE Workshop on Neural Networks for Signal Processing*, 21-29.

Rudin W. (1973). *Functional Analysis*. McGraw Hill, New York.

Scarselli F. and Tsoi A.C. (1998). Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results, *Neural Netw*., 11(1), 15-37.

Shardt, Y. A.W. (2015). *Statistics for Chemical and Process Engineers: A Modern Approach*. Springer International Publishing, Switzerland.

Silva I.N., Spatti D.H., Flauzino R.A., Liboni L. H.B., and Reis-Alves, S.F. (2017). *Artificial Neural Networks: A Practical Course*. Springer International Publishing, Switzerland.

Sprecher D.A. (1965). On the structure of continuous functions of several variables. *Trans. Amer. Math. Soc*., 115, 340-355.

Xu D., Wang B., Zhang G., Wang G. and Yu Y. (2018) A review of sensorless control methods for AC motor drives. *CES Transactions on Electrical Machines and Systems*, 2(1), 104-115.