World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:9, No:7, 2015

Optimization of the Input Layer Structure for Feed-Forward Narx Neural Networks

Zongyan Li, Matt Best

***Abstract*—**This paper presents an optimization method for reducing the number of input channels and the complexity of the feed-forward NARX neural network (NN) without compromising the accuracy of the NN model. By utilizing the correlation analysis method, the most significant regressors are selected to form the input layer of the NN structure. An application of vehicle dynamic model identification is also presented in this paper to demonstrate the optimization technique and the optimal input layer structure and the optimal number of neurons for the neural network is investigated.

***Keywords*—**Correlation analysis, F-ratio, Levenberg-Marquardt, MSE, NARX, neural network, optimisation.

I. Introduction

THIS paper is motivated by work on developing reduced order models for vehicle dynamics using system identification techniques. The idea of the artificial neural networks (ANN), often shortened as neural network, originated from a biological domain. The neural network is a computing system made of simple but highly interconnected elements which process information by their dynamic state response to external inputs. Neural networks have been successfully applied for capturing associations or discovering regularities within a set of patterns where the volume, number of variables or diversity of the data is very great. They also work well in revealing interrelationships which are vaguely understood or difficult to describe adequately with conventional approaches.

The feedforward dynamic NARX (Nonlinear AutoRegressive eXogenous model) models have proven very successful in various engineering applications. In a NARX feedforward neural network, the information moves in only one direction. It enters the network from the input nodes, travels through the hidden layers and produces an overall output. The NARX representation for a general discrete nonlinear system is

$$y(t) = f_s\left(y(t-1), \dots, y(t-n_y), u(t), u(t-1), \dots u(t-n_u)\right) + e(t) \quad (1)$$

where the time-delayed terms model the 'memory' of the dynamic system. $f_s(\cdot)$ is a nonlinear surrogate function of the specific system and *e(t)* is the unexplained noise. A vital task is to find the required number of lagged observations n_y ,n_u in order to generate the auto-regressive structure for the model identification in time series. Many researchers made efforts to optimise the structure of a dynamic neural network in the time-series domain. The fact that any neural network representation for a system can have various solutions of weights can cause difficulty in deciding the number of neurons and the number of layers [1]. Khashei and Bijari [2] described an auto-regressive with integrated moving average (ARIMA) modelling method and improved the accuracy of prediction. The feed-forward NN model has also been utilized in various research areas. Pradhan et al. [3] analysed a landslide-related database and developed a feed-forward neural network model to provide risk assessment. Mohandes et al. [4] used a three-layer feed-forward neural network to predict wind speed. Zhang et al. [5] estimate indoor air contaminants using an optimized neural network.

Various optimisation approaches exist in order to reduce the complexity of the ANN. Performance of the model is usually assessed along with the optimisation process and offers hints for choosing appropriate values for internal coefficients [6], [7]. Taguchi's design of experiments described in [8] also provides a systematic way to reduce complexity. On the other hand, the neurons can also be manipulated by a sequential algorithm starting from an initial infrastructure; the performance of the ANN is assessed by a previously specified criterion and neurons are only added when convergence takes too long or the mean squared error is larger than a pre-defined threshold [9]-[12], [16] utilized a multi-object genetic algorithm (GA) to find the optimal compromise of the ANN structure between performance and complexity. However, the disadvantage is that most of these methods are computationally expensive and difficult to implement [13].

This paper aims to investigate the correlation analysis method of input structure optimisation when developing an ANN model and reduce the order of the dynamic ANN by using selected terms in the input layer. Rather than adopting all or part of the delayed linear terms of inputs and output in the input layer in most NN identification, this approach only selects the most influential regressors within a possible searching range, thus dramatically improves the efficiency of the training process. A neural network model for vehicle dynamics is developed in order to validate the techniques.

II. Input Layer Structure Optimization

In a dynamic system, the I/O of the system for the previous time affects the system output of the current time, which indicates that the dynamic system has 'memory'. A NN model is required to identify to reveal the nonlinear relations between the inputs and outputs of the system and accurately replicate the system dynamics. The input signal values are usually

Zongyan Li and Matt Best are with the aeronautical and engineering department of Loughborough University, UK (e-mail: Z.Li@lboro.ac.uk, M.C.Best@lboro.ac.uk).

normalised into [-1, 1] to facilitate the use of log-sigmoid weighting function.

The inputs for a dynamic neural network are usually formed by a full series of linear time-delayed input terms in ascending order such as *u(t-1), u(t-2), u(t-3)…u(t-na), y(t-1), y(t-2), y(t-3)…,y(t-nb)* according to the designed maximum order of the NN model. Alternatively, a NARX NN input layer can be formed by the selected linear, square and cubic terms which are generated based on the original linear terms. From a regression pool as shown in Fig. 1, the candidate regressors which possess significant dynamic relations with the output can be chosen to form the input layer of the neural network model.

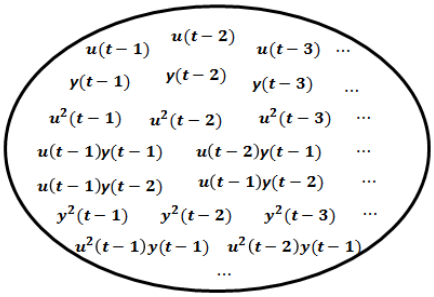

Fig. 1 The pool of candidate regresssor

The former method needs a large number of coefficients and thus increases the computational time of the training process, but saves a lot of time in designing the input layer because the function is already embedded in the Matlab NN toolbox library. However, the latter method offers optimised input terms which carry the most significant system dynamics, thus dramatically reducing network complexity. For selecting the input regressors for the input layer, the correlation analysis method is proposed with the following steps:

A. Correlation Analysis

Firstly, a model regressor pool including all possible candidate regressors is established. For the present technique, all linear, quadratic and cubic regressors with pre-defined maximum time delay forms the overall regressor pool.

At the beginning of the *ith* iteration, the correlation factor between the *jth* regressor $\boldsymbol{x_{j(i)}}$ and dependent output $\boldsymbol{z_i}$ can be represented as:

$$r_{jz} = \sum_{n=1}^{N} \frac{[x_{j(i)}(n) - \bar{x}_{j(i)}][z_i(n) - \bar{z}_i]}{\sqrt{S_{jj}S_{zz}}},\ j = 1,2,3,\ldots, \tag{2}$$

where

$$\bar{x}_j = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x_{j(i)}}(n) \tag{3}$$

$$S_{jj} = \sum_{n=1}^{N} [\boldsymbol{x_{j(i)}}(n) - \bar{x}_{j(i)}]^2 \tag{4}$$

$$S_{zz} = \sum_{n=1}^{N} [\boldsymbol{z_i}(n) - \bar{\boldsymbol{z}}_i]^2 \tag{5}$$

where $\boldsymbol{z_i}$ is defined as the dependent output array which is developed at the beginning of *ith* iteration, $\overline{\boldsymbol{x}}_{j(i)}$ is the mean value of the *jth* regressor vector at the beginning of *ith* iteration. Specifically, $\boldsymbol{z}_1$ is the initial dependent output variable and is assigned as the original output $\mathbf{y}$. The correlation factors between all the candidate regressors and dependent output $\boldsymbol{z_i}$ are determined for subsequent analysis. The regressor inserted into the model should be the one with the highest correlation factor. Initially, the regressor matrix $\boldsymbol{X}_1$ only contains a column of 1s as offset terms and we define $\boldsymbol{x}_{j(1)}$ as vector of original regressors. The $\mathbf{X}$ matrix is updated as:

$$\boldsymbol{X}_1 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{n\times 1} \tag{6}$$

$$\boldsymbol{X}_{i+1} = [\boldsymbol{X}_i, x_{j(1)}] \tag{7}$$

The parameters for the regressors are defined as vector $\boldsymbol{\theta}$. The regressor with the highest correlation factor will have the highest partial F-ratio. Therefore, only one regressor is added into the model in each iteration. With the default offset term in the model, we are able to establish the following two hypotheses:

$$H_0: \theta_1 = \theta_2 = \cdots = \theta_n = 0 \tag{8}$$

$$H_1: \theta_j \neq 0 \tag{9}$$

where H_0 and H_1 are the null hypothesis and alternative hypothesis respectively. The alternative hypothesis indicates that at least one *jth* regressor is inserted into the input layer. In order to decide which hypothesis is accepted, there are three statistical quantities which should be determined:

$$SS_T = \sum_{n=1}^{N}[y(n) - \bar{y}]^2 = \boldsymbol{y}^T\boldsymbol{y} - N\bar{y}^2 \tag{10}$$

$$SS_R = \sum_{n=1}^{N}[\hat{\boldsymbol{y}}(n) - \bar{y}]^2 \tag{11}$$

$$SS_E = \sum_{n=1}^{N}[\boldsymbol{y}(n) - \hat{y}(n)]^2 = \boldsymbol{y}^T\boldsymbol{y} - \hat{\boldsymbol{\theta}}_i\boldsymbol{X_i}^T\boldsymbol{y} \tag{12}$$

where N is the number of sample points of the regressor vector and $\hat{\boldsymbol{y}}(n)$ is the estimated output computed by $\hat{\boldsymbol{y}} = \hat{\boldsymbol{\theta}}_i\boldsymbol{X_i}^T$ from the model. $\bar{y}$ is the mean value of the output variable. SS_T is the total sum of the squares, SS_R represents the regression sum of squares and SS_E is the residue sum of squares. The three assessment terms are then related as:

$$SS_T = SS_R + SS_E \tag{13}$$

If we substitute (10)-(12) into (13), then the following relation is derived:

$$SS_R = \hat{\boldsymbol{\theta}}_i\boldsymbol{X_i}^T\boldsymbol{y} - N\bar{y}^2 \tag{14}$$

where

$$\hat{\boldsymbol{\theta}}_i = (\boldsymbol{X_i}^T\boldsymbol{X_i})^{-1}\boldsymbol{X_i}^T\boldsymbol{y} \tag{15}$$

Subsequently, to assess the regressors, two statistical processes are considered in advance of adding each regressor into the input layer.

B. Forward Selection

The partial F-ratio decides the significance of the regressor. In the situation that the model already contains *m* regressors, the *jth* regressor can be brought into the input layer if

$$F = \frac{SS_R(\hat{\theta}_j|\hat{\theta}_m)}{s^2} = \frac{SS_R(\hat{\theta}_{m+j}) - SS_R(\hat{\theta}_m)}{s^2} > F_{in} \quad (16)$$

where

$$SS_R(\hat{\theta}_{m+j}) = \hat{\theta}_{m+j} X_{m+j}^T y - N\bar{y}^2 \quad (17)$$

$$s^2 = \frac{1}{N-(m+1)} * (y - \hat{\theta}_{m+j} X_{m+j}^T)^T (y - \hat{\theta}_{m+j} X_{m+j}^T) \quad (18)$$

$SS_R(\hat{\theta}_{m+j})$ is the regression sum of squares obtained by adding the *jth* regressor to the original m terms. The index $m+j$ generally means the *jth* regressor is added to the original m terms in the input layer. s^2 is the residual sum of squares after the jth regressor is added into the structure.

C. Backward Elimination

The regressors already entered in the input layer are reassessed by means of their partial F-ratios in each iteration, since a regressor added in the input layer at the early stage may become redundant when it involves some relationship with the regressors added subsequently. With the input design that already involves p regressors, the jth regressor with the lowest partial F-ratio is eliminated if

$$F = \min_j \frac{SS_R(\hat{\theta}_p) - SS_R(\hat{\theta}_{p-j})}{s^2} < F_{out} \quad (19)$$

where

$$SS_R(\hat{\theta}_{p-j}) = \hat{\theta}_{p-j} X_{p-j}^T y - N\bar{y}^2 \quad (20)$$

and $SS_R(\hat{\theta}_{p-j})$ is the regression sum of squares obtained by removing the *jth* regressor from the *p* terms which are already in the model. The index $p-j$ generally represents the *jth* regressor being removed from the original input layer containing *p* terms.

D. Iterative Structure Update

Finally, the structure of the input layer is represented as the regressors included in the **X** matrix. In order to remove the influence of the selected regressors, the dependent output variable z_i and candidate regressors which are not selected are modified according to the regressors already in the input layer design, i.e., at the end of the *ith* iteration, the dependent variable is altered as

$$z_{i+1} = y - X_i \hat{\theta}_i \quad (21)$$

and all the remaining regressors modified by removing the least squares components formed by already selected terms and the next iteration becomes

$$x_{j(i+1)} = x_{j(i)} - X_i \hat{\beta}_{j(i)},\ j = 1,2,3 \ldots \quad (22)$$

where

$$\hat{\beta}_{j(i)} = (X_i^T X_i)^{-1} X_i x_{j(i)},\ j = 1,2,3 \ldots \quad (23)$$

At the end of this iteration, *i* is increased by 1 for the next iteration.

The iterations from step A to step D continue until no other candidate regressor in the regressor pool possesses a partial F-ratio higher than F_{in} and no regressor in the model has a partial F-ratio less then F_{out}, where F_{in} and F_{out} are the preselected stopping criteria for the iteration. At 95% confidence level, we use the criterion F(0.05, 1, N −m) ≈ 4, where the sample number N is much larger than number of the identified coefficients *m*. In other words, if the selected regressor possesses a partial F-ratio larger than 4, there is at least 95% chance that we made the correct decision to add the regressor into the input layer. Usually we find $F_{in} = F_{out}$, however $F_{in} > F_{out}$ indicates it is harder to accept a regressor than delete one. As a result, all the significant terms in the regressor pool are found and inserted into the input layer through the iteration process. The selection process stops when the number of qualified regressors has reached the pre-defined maximum in the input layer or there are no further qualified regressors to be selected.

III. Setup of a Two Layer Network

The two layer neural network structure comprises a hidden layer and an output layer. In each neuron of the hidden layer, a threshold function is defined and the neuron is 'fired' when the weighted sum of inputs reaches a particular threshold. In the example, the log-sigmoid function which generates a threshold at 0 and +1 is used. For most nonlinear problems, one hidden layer is sufficient to recognise continuous wave pattern and the number of neurons needed in the hidden layer is one of the key parameters in defining the complexity of the NN model. By building NN models with increasing number of neurons and comparing validation results, the optimal number of neurons can be determined. Although a computationally expensive process, the searching can be done effectively and automatically. The final delivered neural network model does not need to run this process again. In Fig. 2, in the input layer, *x(t)* with the number '4' underneath indicates that 4 input channels are formed by the input regressors including linear and nonlinear terms. The output signal *y(t)* goes through a delay circle marked '1' in the center and becomes the one-step ahead output *y(t-1)*. Apart from *y(t-1)*, the rest of input channels are within the block of *x(t)*. For a two-layer MISO dynamic neural network using log-sigmoid weighting function, the network can be analytically defined as

$$y(t) = \sum_{i=1}^{m} \alpha_i \cdot f_h(\sum_{k=1}^{n} w_{ik} x_k(t) + b_{ik}) + b_0 \quad (24)$$

where *m* is the number of neurons in the hidden layer, *n* is the number of input nodes, f_h is the weighting function used in the hidden layer, w_{ik} and b_{ik} the weight and bias corresponding to

the *kth* input in the *ith* neuron. α_i and b_0 are the weights and bias in the output layer.

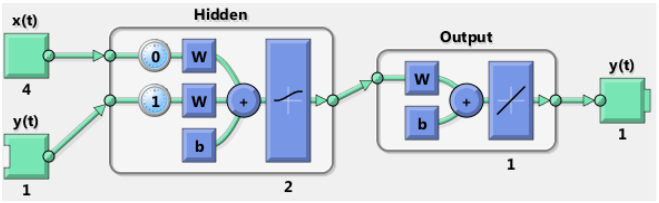

Fig. 2 Two layer perceptron networks

Hence, the weighted sum of the weighted inputs within the hidden layer can be represented as:

$$\boldsymbol{S} = \sum_{k=1}^{n} w_{ik}\boldsymbol{x}_k(t) + b_{ik} \tag{25}$$

$$= \begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdots & w_{1n} \\ w_{21} & w_{22} & w_{23} & \cdots & w_{2n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ w_{m1} & w_{m2} & w_{m3} & \cdots & w_{mn} \end{bmatrix}_{m\times n} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ \vdots \\ b_{1m} \end{bmatrix}$$

$$= \begin{bmatrix} b_{11} + w_{11}x_1 + w_{12}x_2 + \cdots + w_{1n}x_n \\ b_{12} + w_{21}x_1 + w_{22}x_2 + \cdots + w_{2n}x_n \\ \vdots \\ b_{1m} + w_{m1}x_1 + w_{m2}x_2 + \cdots + w_{mn}x_n \end{bmatrix} \tag{26}$$

Then this weighted sum of the inputs $\boldsymbol{S}$ is applied to the log-sigmoid weighting function which determines which neurons are excited by calculating:

$$h = f_h(\boldsymbol{S}) = \frac{1}{1+\exp(\boldsymbol{S})}, 1 > h > 0 \tag{27}$$

the first derivative of the log –sigmoid function *h* with respect to the weights and bias used in the training algorithm is

$$\frac{\partial h}{\partial w_{ik}} = -\frac{\exp(\boldsymbol{S})}{[1+\exp(\boldsymbol{S})]^2} \cdot \boldsymbol{x}_k \tag{28}$$

$$\frac{\partial h}{\partial b_{ik}} = -\frac{\exp(\boldsymbol{S})}{[1+\exp(\boldsymbol{S})]^2} \tag{29}$$

The output layer then uses the linear transfer function to filter the weighted sum of output from the hidden layer. The output from each neuron of the hidden layer is linearly gained and biased without changing the nonlinear dynamics.

For the training process, at least two sets of data are commonly used: training data for establishing the network and test data for validation. The weights are initially assigned randomly and the training process is supervised by the measured output of the training data. In order to determine coefficients $\boldsymbol{\theta} = [\boldsymbol{w}, \boldsymbol{b}]$ in the neural network, the Levenberg-Marquardt (LM) algorithm [14], [15], also known as Nonlinear Least Squares Minimisation, is used. The problem for the application of LM algorithm is defined as optimizing **,** so that the sum of squares of the errors:

$$L(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N}[(y_i - f(x_i, \boldsymbol{\theta})]^2 = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} \tag{30}$$

is minimised. N is the number of input and output samples. The index i represents the sample number for a pair of input and output. $f(x_i, \theta)$ is a non-linear function which estimates the output, in this case, the neural network model.

The LMA is an iterative process starting with an initial guess of the coefficient. In each epoch the coefficient $\boldsymbol{\theta}$ is varied by a small amount $\boldsymbol{\delta}$, hence a new estimation $f(x_i, \boldsymbol{\theta} + \boldsymbol{\delta})$ is calculated as:

$$f(x_i, \boldsymbol{\theta} + \boldsymbol{\delta}) \approx f(x_i, \boldsymbol{\theta}) + J_i\boldsymbol{\delta} \tag{31}$$

where J_i is the back-propagation gradient defined as:

$$J_i = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\theta}} = \frac{\partial[y_i - f(x_i, \boldsymbol{\theta})]}{\partial \boldsymbol{\theta}} = -\frac{f(x_i, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{32}$$

Hence the gradient for weight and bias are

$$J_{iw} = -\alpha_i \frac{\partial h}{\partial \boldsymbol{w}} \tag{33}$$

$$J_{ib} = -\frac{\partial h}{\partial \boldsymbol{b}} \tag{34}$$

Therefore, the following equation holds:

$$L(\boldsymbol{\theta} + \boldsymbol{\delta}) \approx \frac{1}{2}\sum_{i=1}^{N}(y_i - f(x_i, \boldsymbol{\theta}) - J_i\boldsymbol{\delta})^2 \tag{35}$$

$$= \frac{1}{2}||\boldsymbol{y} - f(\boldsymbol{\theta}) - \boldsymbol{J\delta}||^2 \tag{36}$$

where $\boldsymbol{J}$ is the Jacobian matrix which contains the first derivatives of the network errors with respect to the weights and bias. J_i is the *ith* row of $\boldsymbol{J}$, $f(x_i, \boldsymbol{\theta})$ is the *ith* row of $f(\boldsymbol{\theta})$ and y_i is the *ith* row of $\boldsymbol{y}$.

Taking the derivative of (28) and setting the result to zero leads to:

$$(\boldsymbol{J^TJ})\boldsymbol{\delta} = \boldsymbol{J^T}[\boldsymbol{y} - f(\boldsymbol{\theta})] \tag{37}$$

Levenberg modifies an adaptive value μ which creates a 'damped' version:

$$(\boldsymbol{J^TJ} + \mu\boldsymbol{I})\boldsymbol{\delta} = \boldsymbol{J^T}[\boldsymbol{y} - f(\boldsymbol{\theta})] \tag{38}$$

The damping factor μ is adjusted in each iteration according to the convergence speed. A smaller μ can be used when the convergence speed is rapid.

Therefore, the increment of coefficient $\boldsymbol{\delta}$ can be determined as

$$\boldsymbol{\delta} = \boldsymbol{\theta_{k+1}} - \boldsymbol{\theta_k} = (\boldsymbol{J^TJ} + \mu\boldsymbol{I})^{-1}\boldsymbol{J^T}[\boldsymbol{y} - f(\boldsymbol{\theta_k})] \tag{39}$$

The LM backpropagation is achieved by performing the gradient descent within the solution's vector space towards a 'global minimum'. The LM algorithm appears to be the fastest method for training moderate-sized feedforward neural networks (up to several hundred weights) [16]. Moreover, this method uses the Jacobian for calculations, which assumes that performance is measured by a mean or sum of squared errors.

IV. Application in Dynamic Vehicle Modeling

A Jaguar Land Rover SUV is used to test the identification of a reduced order dynamic model. The dynamic information of the vehicle is generated by a high-fidelity model in CarMaker. As illustrated in Fig. 3, a virtual vehicle ride test is conducted on a randomly generated digital road and three segments of data which represents the dynamics of the SUV operating at 10m/s, 20m/s and 30m/s respectively. The random road profile approximates a nominally high real world conditions, and the test data are collected at sampling rate 100Hz.

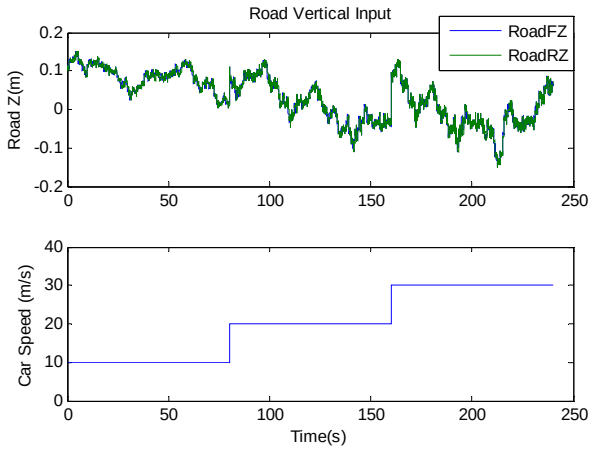

Fig. 3 Inputs of virtual test for NN identification (RoadFZ: road displacement at front wheel, Road RZ: road displacement at back wheel)

A two layer dynamic neural network structure is established between the road input and vehicle dynamic response represented by various dynamic outputs (pitch angle as an example). The original linear inputs are shown in Table. I, and the identified neural network model with various reduced input sets illustrate the optimization process shown in Table. II. The regressor is selected by assessing its correlation and partial F-ratio. The regressor with the highest correlation would be found at each iteration and every newly added regressor can rebalance the partial F-ratio for all the selected regressors.

Final results are given in Table. III revealing that the training time is reduced significantly because of reduced number of weights when the neuron number is optimized. In order to assess and compare quality, the model performance is measured by MSE (Mean Squared Error) and R-square:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 \tag{40}$$

$$R^2 = \frac{e^T e}{Y^T Y} \times 100 \tag{41}$$

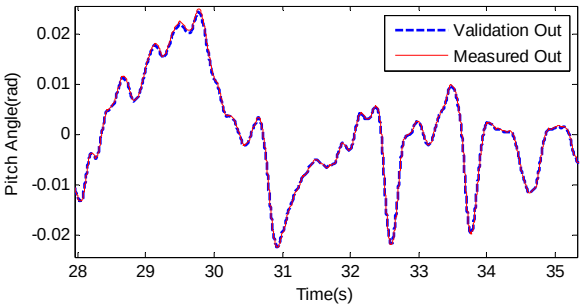

Fig. 4 Validation of the optimized NN model for vehicle ride dynamics

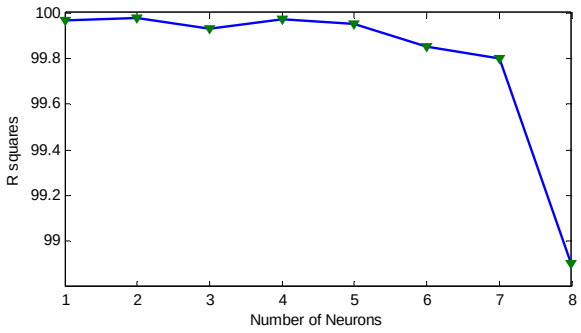

Fig. 5 Search for the optimal neuron number for the optimized neural network

Validation results showed that the optimized NN model is able to achieve a slightly higher value of R^2 while keeping a lower number of weights. Fig. 4 demonstrates this result for the vehicle ride model and proves the effectiveness of the optimized neural network. Based on the optimized input layer structure including five selected terms shown in Table II, we continue to search for the optimal number of neurons which can produce the model with best quality whereas this value is kept as small as possible in order to reduce the possibility of over-fit. Fig. 5 reveals the variation of R^2 in validation result along with increase of neuron number and this illustrate that model quality would not necessarily keep improving due to the side effect of over-fitting which decreases model accuracy. Therefore an optimal number of neurons can be achieved by searching for the best compromise between model complexity and accuracy. For this specific problem, the optimal neuron number is chosen as 2.

TABLE I
Definition of Inputs and Outputs for Vehicle Ride Model

Original Inputs/Outputs	Basic Regressors for the Input Layer	Units
Vertical Road Displacement for Front Wheel	$u_1(t-1), u_1(t-2), u_1(t-3), u_1(t-4), u_1(t-5), u_1(t-6)$	m
Vertical Road Displacement for Rear Wheel	$u_2(t-1), u_2(t-2), u_2(t-3), u_2(t-4), u_2(t-5), u_2(t-6)$	m
Vehicle Forward Velocity	$u_3(t-1), u_3(t-2), u_3(t-3), u_3(t-4), u_3(t-5), u_3(t-6)$	m/s
Pitch Angle	$y(t-1), y(t-2), y(t-3), y(t-4), y(t-5), y(t-6)$	rad

TABLE II
CORRELATION ANALYSIS FOR FIVE SELECTED INPUTS

Iteration No.	1	2	3	4	5
Correlation Factor	0.9946	0.9636	0.0887	0.1153	0.0441
Regressors added at each iteration	Partial F-ratio				
$y(t-1)$	2206800	1281200	1290800	1307100	1308900
$y(t-2)$		311900	314500	294570	295020
$u_3^2(t-6)$			190	376	411
$u_3(t-2) \times y(t-2)$				323	327
$u_1^2(t-1) \times u_2(t-1)$					47

TABLE III
COMPARISON BETWEEN FULL LINEAR TIME-SERIES INPUTS AND OPTIMIZED INPUTS

Input design	Input term number	Neuron number/ Weights number	MSE (10^{-7})	R^2	Training time (second)	Simulation time (second)
Full Linear time-series	24	2 / 50	4.57	99.61	18.927	2.59
Optimized inputs	5	2 / 12	0.251	99.98	7.081	2.49

V. CONCLUSION

In this paper, the input layer structure of the NARX neural network is formed from regressors selected using a correlation analysis method. The technique has been successfully implemented in the identification process of vehicle ride model and the optimal neuron number and input layer structure are determined for the developed neural network. It can be concluded that the input layer optimization process has successfully reduced the computational time for neural network training whereas the model validation accuracy is maintained to a high standard.

ACKNOWLEDGMENTS

This work was supported by Jaguar Land Rover and the UK-EPSRC grant EP/xxxxxxx/x as part of the jointly funded Programme for Simulation Innovation.

REFERENCES

[1] D. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms". *Proc.1989 International Joint Conf. Artificial Intelligence*.

[2] M. Khashei and M. Bijari, "An artificial neural network (p, d,q) model for timeseries forecasting". *Expert Systems with Applications* Vol. 37 ,2010, pp. 479–489

[3] B. Pradhan and S. Lee and M. F. Buchroithner, "A GIS-based back-propagation neural network model and its cross-application and validation for landslide susceptibility analyses". *Computers, Environment and Urban Systems*, Vol 34, 2010, pp. 216-235

[4] M.A. Mohandes and S. Rehman and T.O. Halawani, "A neural networks approach for wind speed prediction". *Renewable Energy*, Vol. 13,No. 3, 1998, pp.345-354

[5] L. Zhang and F. Tian and S. Liu etc, "Chaos based neural network optimization for concentration estimation of indoor air contaminants by an electronic nose". *Sensors and Actuators A*, Vol. 189, 2013, pp. 161-167.

[6] P. G. Benardos, and G. C. Vosniakos, "Prediction of surface roughness in CNC face milling using neural networks and Taguchi's design of experiments" *Robotics and Computer Integrated Manufacturing*, Vol. 18, 2002, pp. 43–354.

[7] L. Ma, and K. Khorasani, "A new strategy for adaptively constructing multilayer feed-forward neural networks". Neurocomputing, 51, 2003, pp. 361–385.

[8] J. P Ross, "Taguchi techniques for quality engineering". New York: McGraw-Hill, 1996.

[9] S. D. Balkin and J. K. Ord, "Automatic neural network modeling for univariate time series". *International Journal of Forecasting*, Vol. 16, 2000, pp. 509–515

[10] M. M. Islam and K. Murase, "A new algorithm to design compact two hidden layer artificial neural networks". *Neural Networks*, 14, 2001, pp.1265–1278.

[11] X. Jiang and A.H.K.S. Wah, "Constructing and training feed-forward neural networks for pattern classification". *Pattern Recognition* Vol. 36, 2003, pp.853–867.

[12] P. G. Benardos and G. C. Vosniakos, "Optimizing feed-forward artificial neural network architecture". *Engineering Applications of Artificial Intelligence*, 20, 2007, pp. 365–382.

[13] G. Zhang and B. E. Patuwo and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art", *International Journal of Forecasting*, Vol. 14, Issue 1, March, 1998, pp 35–62.

[14] D. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". *SIAM Journal on Applied Mathematics*, Vol. 11, No. 2, June 1963, pp. 431–441.

[15] M. T. Hagan and M. Menhaj, "Training feed-forward networks with the Marquardt algorithm", *IEEE Transactions on Neural Networks*, Vol. 5, No. 6,1994, pp. 989–993.

[16] D. Whitley and T. Starkweather and C. Bogart, "Genetic algorithms and neural networks: optimizing connections and connectivity". *Parallel Computing* Vol.14, 1990, pp. 347-361