Structure Optimisation of Input Layer for Feed-forward NARX Neural Network

Zongyan Li, Loughborough University
Matt Best, Loughborough University

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. Applications of vehicle handling and ride model identification are presented in this paper to demonstrate the optimization technique and the optimal input layer structure and the optimal number of neurons for the NN models are investigated. The results show that the developed NN model requires significantly less coefficients and training time while maintains high simulation accuracy compared with that of the unoptimised model**.**

Keywords: optimisation, correlation analysis, NARX, neural network, F-ratio, MSE, Levenberg-Marquardt.

NOMENCLATURE

Symbol	Quantity
Structure Selection of Input Layer	
$\boldsymbol{y}$	original measured output
$\bar{y}$	mean value of $\boldsymbol{y}$
$\boldsymbol{z}_i$	dependent output- Modified output calculated at the beginning of the *ith* iteration
r_{jz}	correlation factor between *jth* regressor and dependent output **z**
$\boldsymbol{x}_{\boldsymbol{j}(\boldsymbol{i})}$	jth regressor vector at the beginning of *ith* iteration
$\bar{x}_j$	mean value of *jth* regressor vector
S_{jj}	variance of the *jth* regressor
S_{zz}	variance of the dependent output $\boldsymbol{z}_i$
$\boldsymbol{X}_i$	matrix containing the selected regressors of the input layer structure at the begining of the *ith* iteration
SS_T	total sum of the squares
$\widehat{\boldsymbol{\theta}}_{\boldsymbol{i}}$	least squares estimator for the *ith* iteration

$\widehat{\boldsymbol{\theta}}_{\boldsymbol{m+j}}$	least squares estimator when the jth regressor is added to the structure already containing m regressors
$\widehat{\boldsymbol{\theta}}_{\boldsymbol{p-j}}$	least squares estimator when the jth regressor is removed from the structure already containing p regressors
s^2	residual sum of squares based on $\widehat{\boldsymbol{\theta}}_{\boldsymbol{m+j}}$ or $\widehat{\boldsymbol{\theta}}_{\boldsymbol{p-j}}$
$\widehat{\boldsymbol{\beta}}_{\boldsymbol{j(i)}}$	least squares estimator for the *jth* candidate regressor for the *ith* iteration
SS_R	regression sum of squares
SS_E	residue sum of squares
N	number of sample points
n	Index of the input and output vector
Training of Neural Networks	
L	sum of squares of the errors
$\boldsymbol{\alpha}_i$	weights of the linear output layer corresponding to the *ith* neuron
$\boldsymbol{w}_{ik}$	weights of the nonlinear (log-sigmoid) hidden layer corresponding to the *kth* input of the *ith* neuron
$\boldsymbol{b}_{ik}$	bias of the nonlinear (log-sigmoid) hidden layer corresponding to the *kth* input of the *ith* neuron
$\boldsymbol{b}_0$	bias of the linear output layer corresponding to the output
$\boldsymbol{x}_k$	*kth* input vector
$\boldsymbol{S}$	output of the nonlinear hidden layer
J_{iw}	weights in the ith row of the NN Jacobian matrix
J_{ib}	wias in the ith row of the NN Jacobian matrix
$\boldsymbol{\theta}$	vector of coefficients including all weights and bias of the neural network
μ	damping factor of Levenberg-Marquardt algorithm

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 (Susitra and Paramasivam, 2014). They also work well in revealing interrelationships which are vaguely understood or difficult to describe adequately with conventional approaches.

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

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

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 solution of weights can cause difficulty in deciding the number of neurons and the number of layers (Montana and Davis, 1989). An auto-regressive is described with integrated moving average (ARIMA) modelling method and improved the accuracy of prediction (Khashei and Bijari, 2010). These challenges motivated researchers to explore the optimised structure of the NN network and applied them to various engineering project.

Various optimisation approaches exist in order to reduce the complexity of the ANN and offer hints for choosing appropriate values for internal coefficients. Performance of the model is usually assessed along with the optimisation process by evaluating the gradient between

the simulation output error and the change of the weighting coefficients in the nodes of the NN network (Benardos, and Vosniakos, 2002; Ma and Khorasani, 2003). Taguchi's design of experiments, which aims to find the cause and effect between input and output prior to model development, also provides a systematic way to reduce complexity (Ross, 1996). 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 (Balkin and Ord, 2000; Islam and Murase, 2001; Jiang and Wah, 2003). Multi-object genetic algorithm(GA) can be utilized where the impact of the number of layers and neurons, the activation function and training algorithm on the network performance are taken into the calculation of objective function (Benardos and Vosniakos, 2007; Whitley et al., 1990). The offspring ANN models are generated from the initial network structure iteratively until the one with the lowest objective value is found. However, the disadvantage is that most of these methods are computationally expensive and could be difficult to implement for certain problems (Zhang et al., 1998), therefore, it is sensible to consider optimisation techniques focusing on some of the key aspects such as the input layer structure.

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. In the literature, structure detection method using orthogonal least squares (Guo et al., 2015) is also capable of producing good nonlinear terms and reducing the order of the input layer. Statistical algorithm based on linear and nonlinear least squares methods (Hera and Morales, 2014) was used to optimise model-based design as well. The correlation analysis method of feed-forward NN model reduces the number of coefficients in the NN model and can be widely used in various area of application. For

example, a landslide-related database was analysed and a feed-forward neural network model can then be developed to provide risk assessment (Pradhan et al., 2010). A three-layer feed-forward neural network was used to predict wind speed (Mohandes et al., 1998). The indoor air contaminants were modelled using a multi-layer neural network (Zhang and Tian, 2013). These applications of dynamic feedforward NN network applied a full series of linear terms in the input layer and some of the linear terms may not be relevant with the model outputs. Subsequently, a large number of terms in the input layer unnecessarily increase the number of coefficients for the training process. Rather than adopting a long series of the delayed linear terms of inputs and output in the input layer, this approach only selects the most influential regressors within a possible searching range, thus dramatically improves the efficiency of the training process. Optimised neural network models for vehicle ride and handling dynamics are developed in order to validate these techniques.

II. Input Layer Structure Optimisation

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 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 $u(t-1), u(t-2), u(t-3)\ldots u(t-n_a), y(t-1), y(t-2), y(t-3)\ldots, y(t-n_b)$ according to the designed maximum order of the NN model. The maximum order of the system is a guess as the start of the process or based on engineering knowledge. At the beginning of the optimization process, the order numbers 'na' and 'nb' are defined as the maximum order the target system could involve to form the pool of all possible candidate terms. The linear, quadratic and cubic terms are the initial range satisfying the complexity of

system and this range can be extended if necessary. The higher order terms can be generated based on the original linear terms. Accordingly, a regression pool can be defined as shown in Fig.1, compiling of the candidate regressors which possess significant dynamic relations with the output. An efficient NN model can then be determined by selecting the most suitable of these regressors into the input layer.

The standard method needs a large number of coefficients and thus increasing 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. The correlation analysis method is proposed to select the input regressors for the input layer with the following steps as shown in Fig.2:

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 the dependent output $\boldsymbol{z_i}$ can be represented as follows:

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

where

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

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

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

where $\boldsymbol{z_i}$ is defined as the dependent output array which is developed at the beginning of *ith* iteration and $\overline{\boldsymbol{x}}_{j(i)}$ is the mean value of the *jth* regressor vector. Specifically, $\boldsymbol{z}_1$ is the initial dependent output variable and is equivalent to the original output **y**. In this step, the correlation factors between all the candidate regressors and the 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 the vector of original regressors. The **X** matrix is updated as:

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

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

The coefficients for the regressors are defined as vector $\boldsymbol{\theta}$. Therefore, only one regressor with the highest correlation factor 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:

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

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

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

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 original measured 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)(11)(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. The regressor with the highest correlation factor will have the highest partial F-ratio. In the situation that the model already contains *m* regressors, the *jth* regressor can be brought into the input layer if the partial F-ratio

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

where

$$SS_R(\widehat{\boldsymbol{\theta}}_{\boldsymbol{m+j}}) = \widehat{\boldsymbol{\theta}}_{\boldsymbol{m+j}} \boldsymbol{X}_{m+j}^T \boldsymbol{y} - N\bar{y}^2 \tag{17}$$

$$s^2 = \frac{1}{N-(m+1)} * \left(\boldsymbol{y} - \widehat{\boldsymbol{\theta}}_{\boldsymbol{m+j}} \boldsymbol{X}_{m+j}^T\right)^T \left(\boldsymbol{y} - \widehat{\boldsymbol{\theta}}_{\boldsymbol{m+j}} \boldsymbol{X}_{m+j}^T\right) \tag{18}$$

and $SS_R(\widehat{\boldsymbol{\theta}}_{\boldsymbol{m+j}})$ 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

$$F = \min_j \frac{SS_R(\widehat{\boldsymbol{\theta}}_{\boldsymbol{p}}) - SS_R(\widehat{\boldsymbol{\theta}}_{\boldsymbol{p-j}})}{s^2} < F_{out} \tag{19}$$

where

$$SS_R(\widehat{\boldsymbol{\theta}}_{\boldsymbol{p-j}}) = \widehat{\boldsymbol{\theta}}_{\boldsymbol{p-j}} \boldsymbol{X}_{p-j}^T \boldsymbol{y} - N\bar{y}^2 \tag{20}$$

and $SS_R(\widehat{\boldsymbol{\theta}}_{\boldsymbol{p-j}})$ 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 $\boldsymbol{z}_i$ 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

$$\boldsymbol{z}_{i+1} = \boldsymbol{y} - \boldsymbol{X}_i\widehat{\boldsymbol{\theta}}_i \tag{21}$$

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

$$\boldsymbol{x}_{\boldsymbol{j(i+1)}} = \boldsymbol{x}_{\boldsymbol{j(i)}} - \boldsymbol{X}_i\widehat{\boldsymbol{\beta}}_{\boldsymbol{j(i)}},\ j = 1,2,3\ \ldots \tag{22}$$

where

$$\widehat{\boldsymbol{\beta}}_{\boldsymbol{j(i)}} = (\boldsymbol{X}_i^T\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i\boldsymbol{x}_{\boldsymbol{j(i)}},\ j = 1,2,3\ \ldots \tag{23}$$

i is the iteration number and j is the regressor index. 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 is 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 modelling nonlinear problems, at least one hidden layer is used to recognise the relationship represented by continuous function; 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 the training process again and can be directly used on validation data. In Fig.3, in the input layer, *x(t)* with the number '4' underneath indicates that 4 input channels are formed by the input regressors including selected 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

$$y(t) = \sum_{i=1}^{m} \boldsymbol{\alpha}_i \cdot f_h\left(\sum_{k=1}^{n} \boldsymbol{w}_{ik} \boldsymbol{x}_k(t) + \boldsymbol{b}_{ik}\right) + \boldsymbol{b}_0 \tag{24}$$

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, $\boldsymbol{w}_{ik}$ and $\boldsymbol{b}_{ik}$ the weight and bias

corresponding to the *kth* input in the *ith* neuron. $\boldsymbol{\alpha}_i$ and $\boldsymbol{b}_0$ are the weights and bias in the output layer.

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

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

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

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

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

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

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

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 (Marquardt, 1963; Hagan and Menhaj, 1994), also known as Nonlinear Least Squares Minimisation, is used. The problem for the application of LM algorithm is defined as optimising $\boldsymbol{\theta}$ **,** 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.

Therefore, the following equation holds

$$L(\boldsymbol{\theta} + \boldsymbol{\delta}) \approx \frac{1}{2}\sum_{i=1}^{N}(y_i - f(x_i, \boldsymbol{\theta}) - J_i\boldsymbol{\delta})^2 \tag{31}$$

$$= \frac{1}{2}\left|\left|\boldsymbol{y} - f(\boldsymbol{\theta}) - \boldsymbol{J\delta}\right|\right|^2 \tag{32}$$

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 equation (32) and setting the result to zero leads to:

$$(\boldsymbol{J^TJ})\boldsymbol{\delta} = \boldsymbol{J^T}[\boldsymbol{y} - f(\boldsymbol{\theta})] \tag{33}$$

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

$$(\boldsymbol{J^TJ} + \mu\boldsymbol{I})\boldsymbol{\delta} = \boldsymbol{J^T}[\boldsymbol{y} - f(\boldsymbol{\theta})] \tag{34}$$

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

$$\boldsymbol{\delta} = \boldsymbol{\theta_{k+1}} - \boldsymbol{\theta_k} = (\boldsymbol{J^T J} + \mu \boldsymbol{I})^{-1} \boldsymbol{J^T} [\boldsymbol{y} - f(\boldsymbol{\theta_k})] \tag{35}$$

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) (Whitley et al., 1990). 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.

A. Ride model

As illustrated in Fig. 4, a virtual vehicle ride test is conducted on a randomly generated digital road and three segments of ID data are collected when the SUV are operating at 10m/s, 20m/s and 30m/s respectively. Validation data is generated based on running the vehicle on a real world measured road (a UK's B class country road). The test data are collected at the sampling rate of 500Hz and then down-sampled into 100Hz.

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 results showed that 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 in the input layer.

The results given in Table. III reveal 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^2 where R^2=100 corresponds to 100% fitness:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 \tag{36}$$

$$R^2 = \frac{e^T e}{Y^T Y} \times 100 \tag{37}$$

Validation results show that the optimized NN model is able to achieve a slightly higher value of R^2 while keeping a lower number of weights. Fig.5 demonstrates validation result of pitch angle and shows 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.6 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.

B. Handling model

By using the same 100Hz sampling rate as that of the ride model test, the second application aims to demonstrate the effectiveness of NN for modeling the vehicle handling dynamics. As shown in Fig.7, the steer input profile contains variations in different frequencies in order to excite a wide spectrum of the vehicle dynamic response. Meanwhile, the vehicle is accelerated and decelerated within different segments of the test in order to generate more nonlinear behavior of the vehicle handling system. Therefore the handling model is designed as a MIMO model with 2 inputs (steer angle, forward velocity) and 6 outputs (lateral acceleration, yaw rate, roll angle, lateral velocity, roll velocity, roll acceleration).

Fig.8 shows the comparison between measured lateral acceleration and the simulated one from the handling model. It demonstrates that the nonlinear behavior of the vehicle handling system can be accurately predicted by NN network. Six relevant dynamic outputs have been modelled using all linear regressors and selected regressors for the input layer separately. The details of the best model accuracy and training time are shown in Table IV. It can be seen that the training time for the NN with selected regressors in the input layer is halved compared with the one with all linear regressors in the input layer. The corresponding coefficients are shown in Table V and Table VI.

V. CONCLUSION

In this paper, the training and input layer optimization technique for a forward NARX NN network has been described. The input layer structure of the NARX neural network is formed from regressors selected by the correlation analysis method. Partial F-ratio analysis has also been applied as a key feature to determine the structure of the input layer structure. The applications of vehicle ride and handling model identification have been presented and minimal neuron number and input layer structure are determined for the developed neural network. The NARX NN network was trained by Levenberg-Marquardt algorithm using

selected regressors in the input layer. 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.

ACKNOWLEGEMENTS

This work was supported by Jaguar Land Rover and the UK-EPSRC grant EP/K014102/1 as part of the jointly funded Programme for Simulation Innovation.

REFERENCES

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

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

Benardos P. G. and Vosniakos G. C. (2007) 'Optimizing feed-forward artificial neural network architecture'. *Engineering Applications of Artificial Intelligence*, Vol. 20, pp. 365–382.

Guo, Y., Guo, L., Billings S.A. and Wei H.L. (2015) 'Identification of nonlinear systems with non-persistent excitation using an iterative forward orthogonal least squares regression algorithm'. *Int. J. of Modelling, Identification and Control*, 2015, Vol.23, No.1, pp.1 – 7

Hagan M. T. and Menhaj M. (1994) 'Training feed-forward networks with the Marquardt algorithm', *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, pp. 989–993.

Hera P.L. and Morales D.O. (2014) 'Non-linear dynamics modelling description for simulating the behaviour of forestry cranes', *Int. J. of Modelling, Identification and Control*, 2014, Vol.21, No.2, pp.125–138.

Islam M. M. and Murase K. (2001) 'A new algorithm to design compact two hidden layer artificial neural networks'. *Neural Networks*, Vol. 14, pp. 1265–1278.

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

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

Ma L. and Khorasani K. (2003) 'A new strategy for adaptively constructing multilayer feed-forward neural networks'. *Neurocomputing*, Vol.51, pp. 361–385.

Marquardt D. (1963), 'An Algorithm for Least-Squares Estimation of Nonlinear Parameters'. *SIAM Journal on Applied Mathematics*, Vol. 11, No. 2, pp. 431–441.

Montana D. and Davis L. (1989) 'Training feedforward neural networks using genetic algorithms'. *Proc. International Joint Conf. Artificial Intelligence*, Vol.1, pp.762-767.

Mohandes M.A., Rehman S. and Halawani T.O. (1998) 'A neural networks approach for wind speed prediction'. *Renewable Energy*, Vol. 13, No.3, pp.345-354

Pradhan B. and Lee S. and Buchroithner M. F. (2010) '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, pp. 216-235

Ross J. P (1996) *Taguchi techniques for quality engineering*. McGraw-Hill, New York.

Susitra D. and Paramasivam S. (2014) 'Artificial intelligence-based rotor position estimation for a 6/4 pole switched reluctance machine from phase inductance'. *Int. J. of Modelling, Identification and Control*, 2014, Vol.22, No.1, pp.68 - 79

Whitley D. and Starkweather T. and Bogart C. (1990) 'Genetic algorithms and neural networks: optimizing connections and connectivity'. *Parallel Computing* Vol. 14, pp. 347-361

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

Zhang L., Tian F. and Liu Set al, (2013) 'Chaos based neural network optimization for concentration estimation of indoor air contaminants by an electronic nose'. *Sensors and Actuators A*, Vol. 189, pp. 161-167.

FIGURES AND TABLES

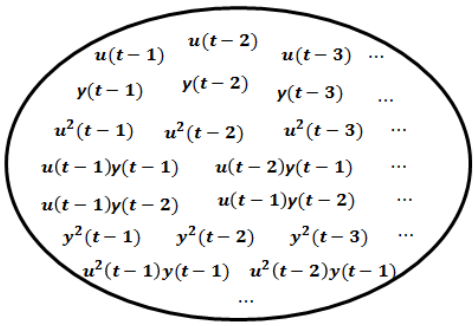

Fig.1 The pool of candidate regresssor

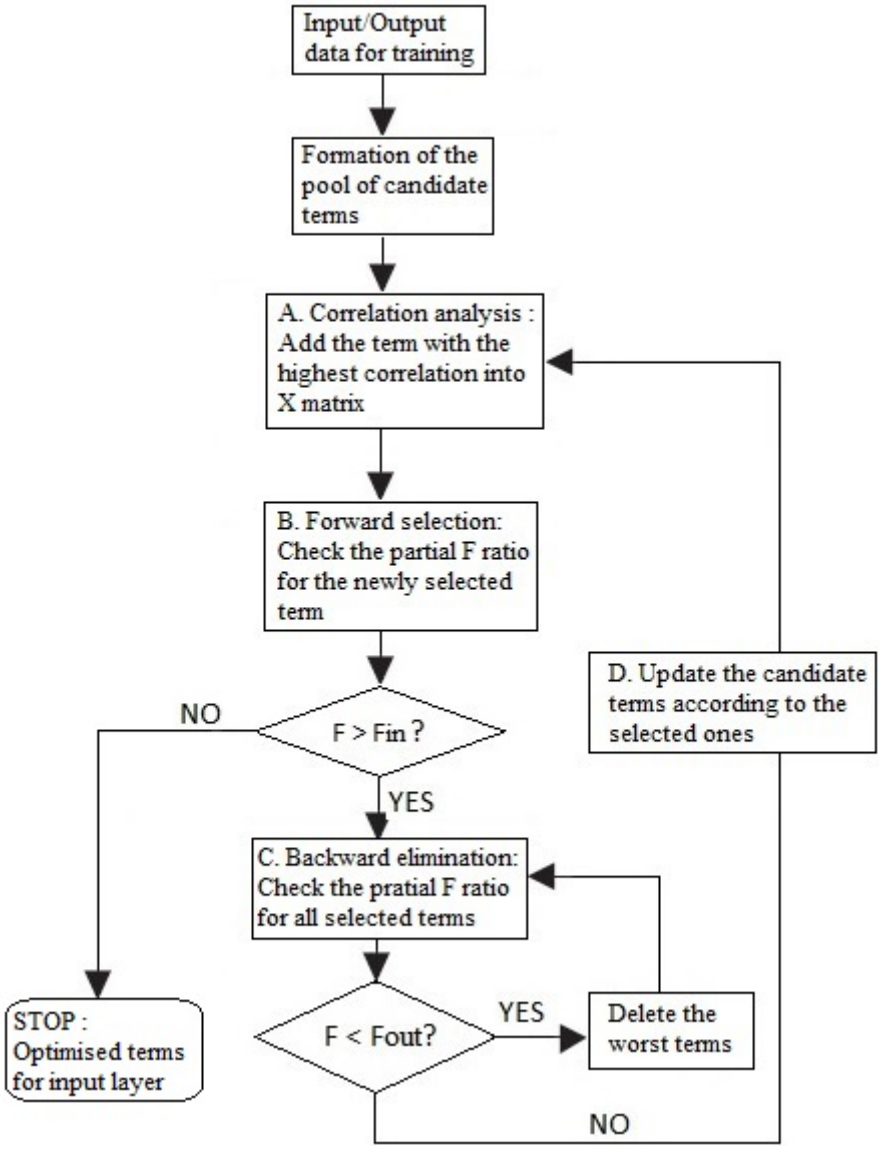

Fig.2 Flowchart for input layer structure selection

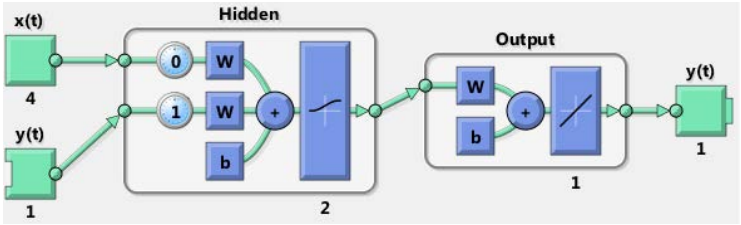

Fig.3 Two layer perceptron networks

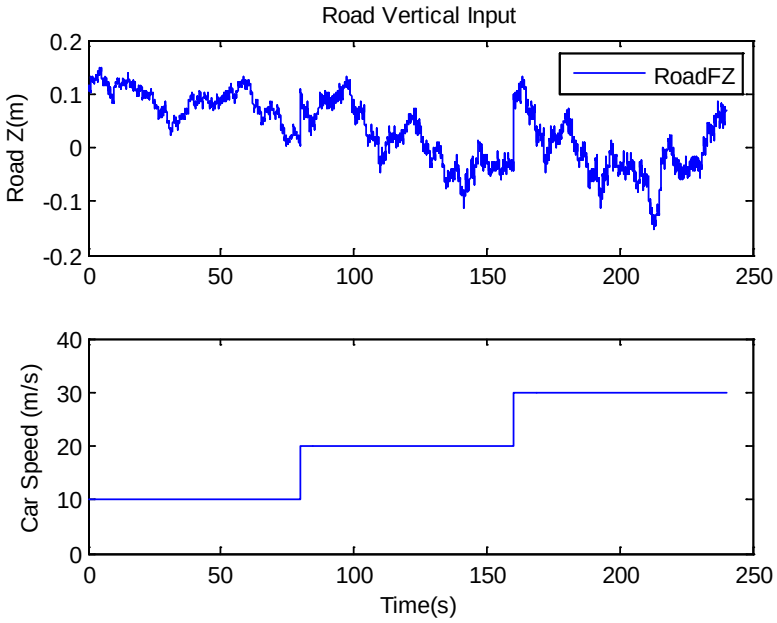

Fig.4 Inputs of virtual test for NN identification (RoadFZ: road displacement at front wheel)

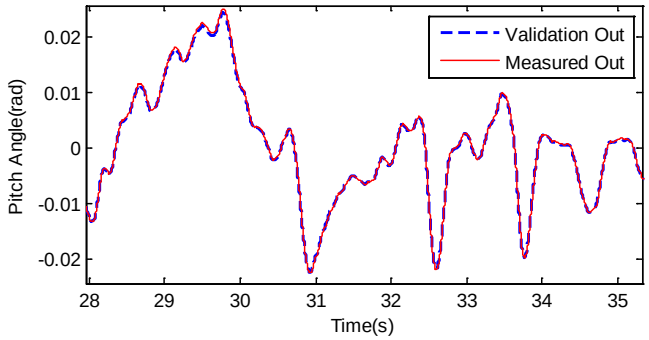

Fig.5 Validation of the optimized NN model for vehicle ride dynamics

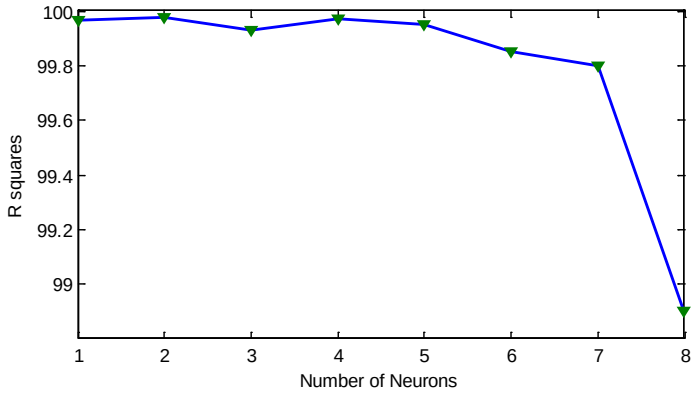

Fig.6 Search for the optimal neuron number for the optimized neural network

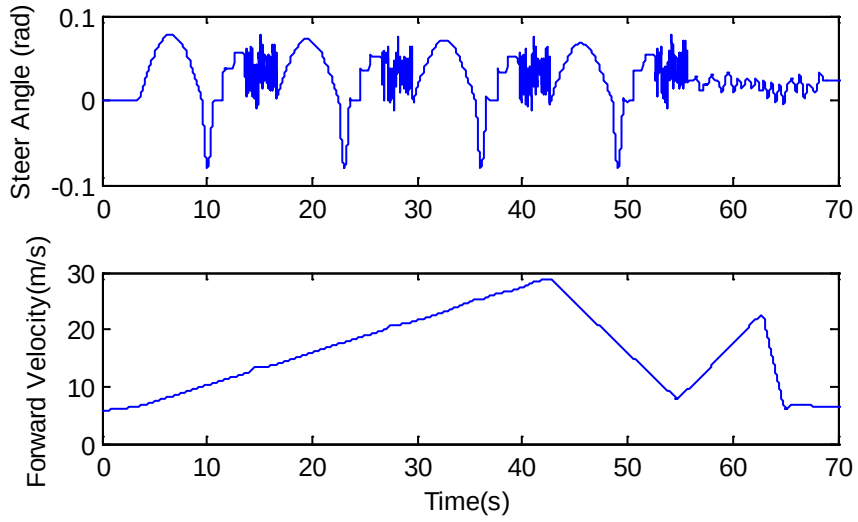

Fig.7 Input profiles for handling model validation (u1: steer angle, u2: forward velocity)

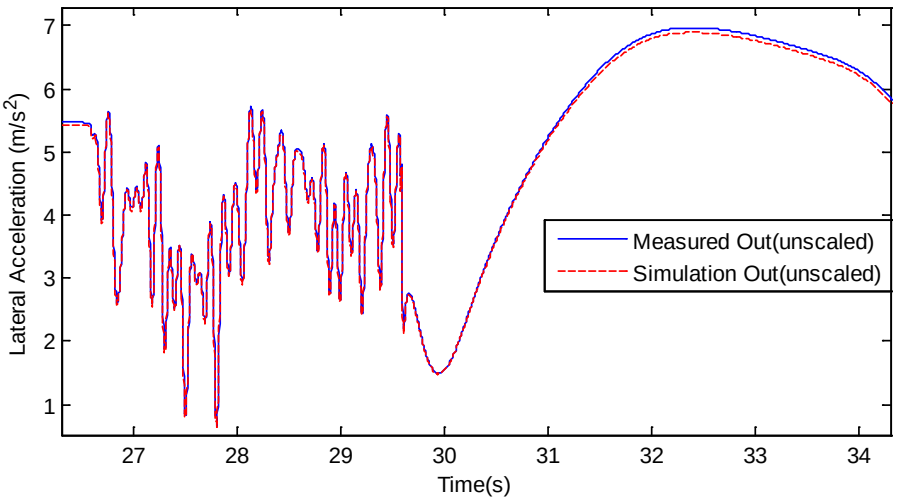

Fig.8 Validation results of handling model

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
Selected Regressors	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 OPTIMISED INPUTS

Input design	Input term number	Neuron number/total number of weights	MSE (10^{-7})	R^2	Training time (second)	Simulation time (s/s)
Full Linear time-series	24	2 / 50	4.57	99.61	18.927	2.59/90
Optimized inputs	5	2 / 12	0.251	99.98	7.081	2.49/90

TABLE IV
OUTPUT VALIDATION RESULTS FOR HANDLING MODEL

Outputs	Units	Model Accuracy (R squares)		Average MISO Training Time/real time (s/s)		Number of coefficients		Average MISO Simulation Time (s/s)
		with all linear regressors for the input layer	with Selected regressors for the input layer	with all linear regressors	with selected regressors	with all linear regressors	with selected regressors	
Lateral Acceleration	m/s^2	99.9890	99.9875	3.189/40	1.558/40	626	96	2.89/90
Yaw Rate	rad/s	99.9984	99.9982					
Roll Angle	rad	99.9993	99.9985					
Lateral Velocity	m/s	99.9995	99.9985					
Roll Velocity	rad/s	99.9833	99.8178					
Roll Acceleration	rad/s^2	99.5124	99.0619					

TABLE V

DEFINITION OF INPUTS AND OUTPUTS FOR VEHICLE HANDLING MODEL

Original Inputs/Outputs	Basic Regressors for the Input Layer	Units
Steer Angle	$u_1(t-1), u_1(t-2), u_1(t-3),$ $u_1(t-4), u_1(t-5), u_1(t-6)$	rad
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
Lateral Acceleration	$y(t-1), y(t-2), y(t-3),$ $y(t-4), y(t-5), y(t-6)$	rad/s^2

TABLE VI

CORRELATION ANALYSIS FOR THE SELECTED INPUTS CHANNELS

Iteration No.	1	2	3	4	5
Correlation Factor	0.9981	0.7243	0.1522	0.0873	0.0858
Selected Regressors	Partial F-ratio				
$y(t-1)$	1156800	27418	14767	12423	12311
$y(t-2)$		4856	1351	676	671
$y(t-6)$			104	134	134
$y(t-3)$				34	33
$u_1^2(t-1)$ $\times u_1(t-6)$					33