Hindawi Publishing Corporation
Journal of Engineering
Volume 2013, Article ID 421543, 9 pages
http://dx.doi.org/10.1155/2013/421543

Research Article

Design a PID Controller for Suspension System by Back Propagation Neural Network

M. Heidari[1] and H. Homaei[2]

[1] *Mechanical Engineering Group, Aligudarz Branch, Islamic Azad University, Aligudarz, Iran*
[2] *Faculty of Engineering, Shahrekord University, Shahrekord, Iran*

Correspondence should be addressed to M. Heidari; moh104337@yahoo.com

Received 14 January 2013; Accepted 10 February 2013

Academic Editor: Jie Zhou

This paper presents a neural network for designing of a PID controller for suspension system. The suspension system, designed as a quarter model, is used to simplify the problem to one-dimensional spring-damper system. In this paper, back propagation neural network (BPN) has been used for determining the gain parameters of a PID controller for suspension system of automotive. The BPN method is found to be the most accurate and quick. The best results were obtained by the BPN by Levenberg-Marquardt algorithm training with 10 neurons in the one hidden layer. Training was continued until the mean squared error is less than $1e-5$. Desired error value was achieved in the BPN, and the BPN was tested with both data used and not used for training. By training of this network, it is possible to estimate the gain parameters of PID controller at any condition. The inputs of network are automotive velocity, overshoot percentage, settling time, and steady state error of suspension system response. Also outputs of the net are the gain parameters of PID controller. Resultant low relative error value of the ANN model indicates the usability of the BPN in this area.

1. Introduction

Vehicle suspension serves as the basic function of isolating passengers and the chassis from the roughness of the road to provide a more comfortable ride. In other words, a very important role of the suspension system is the ride control. Due to developments in the control technology, electronically controlled suspensions have gained more interest. These suspensions have active components controlled by a microprocessor. By using this arrangement, significant achievements in vehicle response can be carried out. Selection of the control method is also important during the design process. The design of vehicle suspension systems is an active research field in which one of the objectives is to improve the passenger's comfort through the vibration reduction of the internal engine and external road disturbances [1–3]. A design of a mixed suspension system (an actuator in tandem with a conventional passive suspension) for the axletree of a road vehicle based on a linear model with 4 degrees of freedom (dof) has been realized in [4]. The authors proposed an optimal control law that was aimed at optimizing the suspension performance while ensuring that the magnitude of the forces generated by the two actuators and the total forces applied between wheel and body never exceeded the given bounds. Neural network (NN) controllers parallel to McPherson strut-type independent suspensions have been realized in [5]. The major advantages of this control method were its success, robust structure and the ability and adaptation of using these types of controllers on vehicles. Hac [6] applied optimal linear preview control on the active suspensions of a quarter car model. An investigation of the variation of vertical vibrations of vehicles using a radial basis neural network (RBNN) has been presented in [7, 8]. The RBNN was employed to predict the desired values of amplitude of acceleration for different road conditions such as concrete, waved stone, block paved, and country roads. The proposed neural system was also tested for different natural frequencies and the ratios of damping. A methodology for the design of active/hybrid car suspension systems with the goal to maximize passenger comfort (minimization of passenger acceleration) was presented by Spentzas and Kanarachos [9]. For this reason, a neural network (NN) controller was

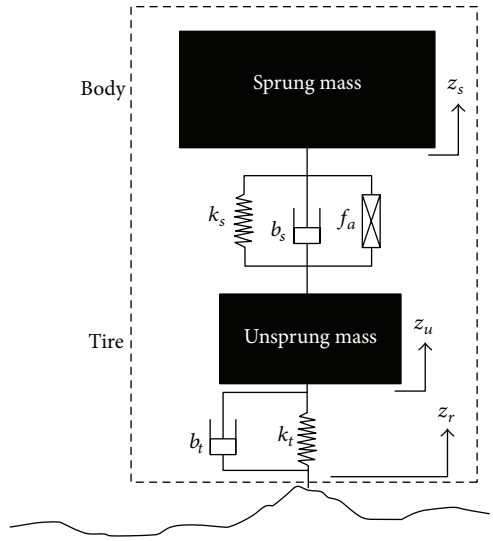

Figure 1: A quarter-car model of suspension system.

proposed, which corresponds to a Taylor series approximation of the (unknown) nonlinear control function and the NN was due to the numerous local minima trained using a semistochastic parameter optimization method. The use of fuzzy-logic-based control for the vehicle-active suspension systems with the two variables going to the fuzzy controller as the active suspension velocity and deflection. The capability of fuzzy logic to model the real-world situations has resulted in its wider application in diverse fields as well. A fuzzy-logic-based control for vehicle-active suspension has been proposed and its capabilities for the improvement of ride comfort and vehicle maneuverability are studied through a software simulation. A control scheme of an active suspension system using a quarter car model has been proposed by Kim and Ro [10]. The authors have shown that due to the presence of nonlinearities such as a hardening spring, a quadratic damping force, and the "tire lift-off" phenomenon in a real suspension system, it was very difficult to achieve the desired performance using linear control techniques. To ensure robustness for a wide range of operating conditions, a sliding mode controller has been designed and compared with an existing nonlinear adaptive control scheme in the literature. The sliding mode scheme utilizes a variant of a sky-hook damper system as a reference model which does not require real-time measurement of road input. A neural scheme for controlling was presented as a bus suspension system. The suspension system, designed as a quarter bus model, was used to simplify the problem to a one-dimensional spring-damper system. The proposed controller was such that the system was always operating in a closed loop, which should lead to better performance characteristics [11].

As seen from previous studies, the researcher used the NN for control of suspension system, but in this study we use the BP neural network to estimate a PID controller. Also the constrains such as overshoot, settling time, and road condition to design a PID controller for suspension system with NN are not examined by other authors. To the authors' best knowledge, no previous studies which cover all these issues are available.

In this paper, a BP neural network was investigated to estimate the gain parameters of PID controller for a suspension system of automotive. The paper is organized in the following manner. Section 2 contains a description of the mathematical model and the problem statement. Section 3 recalls the artificial neural network. Section 4 proposes network development. Simulation results and discussion of the problem are given in Section 5, and finally Section 6 gives the conclusions of this work.

2. Mathematical Model

A quarter-car suspension system shown in Figure 1 is used to simulate the control system. The dynamic equations of the suspension system are of the following form [12]:

$$m_s\ddot{z}_s = -b_s\left(\dot{z}_s - \dot{z}_u\right) - k_s\left(z_s - z_u\right) + f_a \tag{1}$$

$$\begin{aligned} m_u\ddot{z}_u = {} & b_s\left(\dot{z}_s - \dot{z}_u\right) + k_s\left(z_s - z_u\right) \\ & + b_t\left(\dot{z}_r - \dot{z}_u\right) + k_t\left(z_r - z_u\right) - f_a \end{aligned} \tag{2}$$

$$f_a = P_L A_P, \tag{3}$$

where m_s, m_u, k_s, k_t, b_s, and b_t denote the mass, the stiffness, and the damping rate of the sprung and unsprung elements, respectively. Variables z_s, z_u, and z_r are the displacements of body, wheel, and road, respectively.

Also the system is equipped by a hydraulic actuator placed between the sprung and unsprung masses to exert a force f_a between m_s and m_u. P_L and A_P are the fluid pressures in the lower cylinder chamber of the actuator and piston area, respectively. Several points are required to be noted;

(1) The above equations are linearized dynamic equations at equilibrium point and the vehicle speed is constant.

(2) Variables z_s, z_u, and z_r are measured from the static equilibrium position.

(3) The linearized dynamic behavior of tire through interaction with the road is justified where the tire is in contact with the road.

(4) The applied force on tire can be considered as a disturbance force in the system.

Therefore,

$$f_{\text{dis}} = b_t\left(\dot{z}_r - \dot{z}_u\right) + k_t\left(z_r - z_u\right), \tag{4}$$

where f_{dis} is an applied force on the tire from the road.

Equation (2) can be rewritten as

$$m_u\ddot{z}_u = b_s\left(\dot{z}_s - \dot{z}_u\right) + k_s\left(z_s - z_u\right) + f_{\text{dis}} - f_a. \tag{5}$$

Assume that all of the initial conditions are zero, so these equations represent the situation when the wheel goes up a bump. The dynamic equations above can be expressed in a form of transfer functions by taking Laplace transform of the above equations. When the disturbance input z_r was only

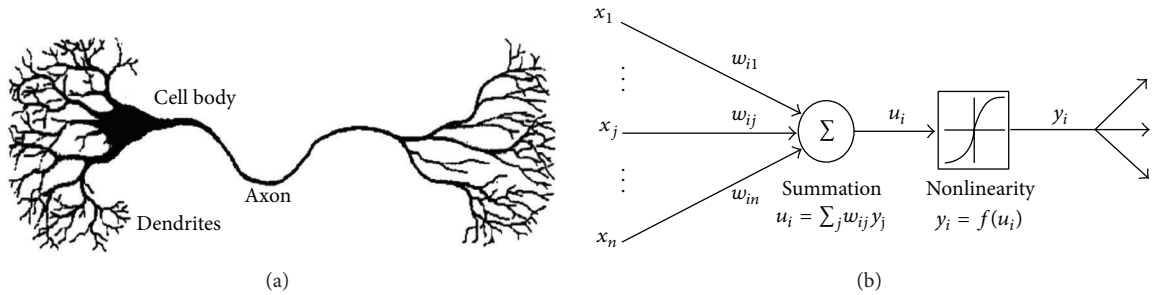

Figure 2: (a) A biological nervous systems and (b) an artificial neuron model.

considered, f_a was set to zero. Thus, the transfer function $G_1(s)$ can be written as follows:

$$G_1(s) = \frac{z_s(s) - z_u(s)}{z_r(s)} = \frac{\left(-m_s b_t s^3 - m_u k_t s^2\right)}{\Delta}$$
$$\Delta = \det \begin{bmatrix} m_s s^2 + b_s s + k_s & -(b_s s + k_s) \\ -(b_s s + k_s) & \left(m_u s^2 + (b_s + b_t) s + (k_s + k_t)\right) \end{bmatrix}. \quad (6)$$

When the control input f_a only was considered, z_r was set to zero. Thus, the transfer function $G_2(s)$ can be written as follows:

$$G_2(s) = \frac{z_s(s) - z_u(s)}{f_a(s)} = \frac{(m_s + m_u) s^2 + b_t s + k_t}{\Delta}. \quad (7)$$

In this context, it is assumed that the car experiences a sinusoidal disturbance from the road, described by the following equation:

$$z_r(t) = (0.01m) \sin(\omega_b t), \quad (8)$$
$$\omega_b = 0.2909v, \quad (9)$$

where v is velocity of car on km/h, ω_b is on rad/sec, and m is total unsprung and sprung mass.

Assuming that each amplitude is completely decoupled and controlled independently from other amplitudes, the control input signal is given by

$$f(t) = K_P e(t) + K_I \int e(t)\, dt + K_D \frac{de(t)}{dt}. \quad (10)$$

In (10), $e(t)$ is the control error

$$e(t) = y_d(t) - y(t), \quad (11)$$

where $y_d(t)$ is the desired car amplitude of displacement and $y(t)$ is the current measured car amplitude. K_P is called the proportional gain, K_I the integral gain, and K_D the derivative gain.

3. Artificial Neural Networks

Artificial NNs are nonlinear mapping systems with a structure loosely based on principles observed in biological nervous systems. In greatly simplified terms as can be seen from Figure 2(a), a typical real neuron has a branching dendritic tree that collects signals from many other neurons in a limited area, a cell body that integrates collected signals and generates a response signal (as well as manages metabolic functions), and a long branching axon that distributes the response through contacts with dendritic trees of many other neurons. The response of each neuron is a relatively simple nonlinear function of its inputs and is largely determined by the strengths of the connections from its inputs. In spite of the relative simplicity of the individual units, systems containing many neurons can generate complex and interesting behaviours.

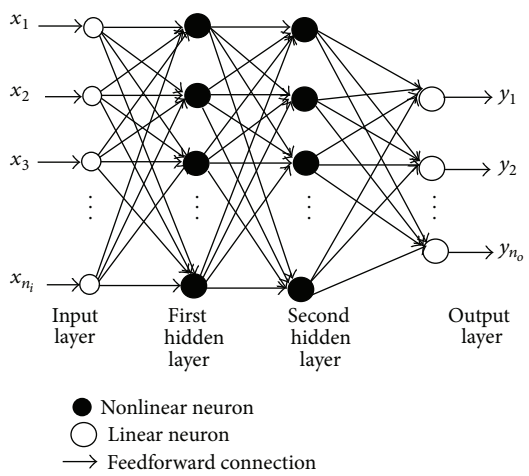

Figure 3: A layered feed-forward artificial NN.

An ANN shown in Figure 3 is very loosely based on these ideas. In the most general terms, an NN consists of a large number of simple processors linked by weighted connections. By analogy, the processing nodes may be called neurons.

Each node output depends only on information that is locally available at the node, either stored internally or arriving via the weighted connections. Each unit receives inputs from many other nodes and transmits its output to other nodes. By itself, a single processing element is not very powerful; it generates a scalar output with a single numerical value, which is a simple nonlinear function of its inputs. The power of the system emerges from the combination of many units in an appropriate way.

A network is specialized to implement different functions by varying the connection topology and the values of the connecting weights. Complex functions can be implemented

by connecting units together with appropriate weights. In fact, it has been shown that a sufficiently large network with an appropriate structure and property chosen weights can approximate with arbitrary accuracy any function satisfying certain broad constraints. Usually, the processing units have responses like (see Figure 2(b))

$$y = f\left(\sum_i u_i\right), \tag{12}$$

where u_i are the output signals of hidden layer to output layer, and $f(u_i)$ is a simple nonlinear function such as the sigmoid or logistic function. This unit computes a weighted linear combination of its inputs and passes this through the nonlinearity to produce a scalar output. In general, it is a bounded nondecreasing nonlinear function; the logistic function is a common choice. This model is, of course, a drastically simplified approximation of real nervous systems. The intent is to capture the major characteristics important in the information processing functions of real networks without varying too much about physical constraints imposed by biology. The impressive advantages of NNs are the capability of solving highly nonlinear and complex problems and the efficiency of processing imprecise and noisy data. Mainly, there are three types of training conditions for NNs, namely, supervised training, graded training, and self-organization training. Supervised training, which is adopted in this study, can be applied as follows.

(1) First, the dataset of the system, including input and output values, is established.

(2) The dataset is normalized according to the algorithm.

(3) Then, the algorithm is run.

(4) Finally, the desired output values corresponding to the input used in test phase.

3.1. Back Propagation Neural Network. Back propagation neural network (BPN), developed by Rumelhart et al. [13], is the most prevalent of the supervised learning models of ANN. BPN used the gradient steepest descent method to correct the weight of the interconnectivity neuron. BPN easily solved the interaction of processing elements by adding hidden layers. In the learning process BPN, the interconnective weights are adjusted using an error convergence technique to obtain a desired output for a given input. In general, the error at the output layer in the BPN model propagates backward to the input layer through the hidden layer in the network to obtain the final desired output. The gradient descent method is utilized to calculate the weight of the network and adjusts the weight of interconnectives to minimize the output error. The formulas used in this algorithm are as follows.

(1) Hidden layer calculation results are

$$\begin{aligned} \text{net}_i &= \sum x_i w_i \\ y_i &= f\left(\text{net}_i\right), \end{aligned} \tag{13}$$

where x_i and w_i are input data and weights of the input data, respectively. f is activation function, and y_i is the result obtained from hidden layer.

(2) Output layer calculation results are

$$\begin{aligned} \text{net}_k &= \sum y_i w_{jk} \\ o_k &= f\left(\text{net}_k\right), \end{aligned} \tag{14}$$

where w_{jk} are weights of output layer, and o_k is the result obtained from output layer.

(3) Activation functions used in layers are logsig, tansig, and linear as

$$\begin{aligned} f\left(\text{net}_i\right) &= \frac{1}{1+e^{-\text{net}_i}} \quad (\text{logsig}) \\ f\left(\text{net}_i\right) &= \frac{1-e^{-\text{net}_i}}{1+e^{-\text{net}_i}} \quad (\text{tansig}) \\ f\left(\text{net}_i\right) &= \text{net}_i \quad (\text{linear}). \end{aligned} \tag{15}$$

(4) Errors made at the end of one cycle are

$$\begin{aligned} e_k &= \left(t_k - o_k\right) o_k \left(1 - o_k\right) \\ e_i &= y_i \left(1 - y_i\right) \sum e_k w_{ij}, \end{aligned} \tag{16}$$

where t_k is result expected from output layer, e_k is error occurred at output layer, and e_i is error occurred at hidden layer.

(5) Weights can be changed using these calculated error values according to (17) as

$$\begin{aligned} w_{jk} &= w_{jk} + \alpha e_k y_i + \beta \Delta w_{jk} \\ w_{ij} &= w_{ij} + \alpha e_i x_i + \beta \Delta w_{ij}, \end{aligned} \tag{17}$$

where w_{ij} are weights of output layer. Δw_{jk} and Δw_{ij} are correction made in weights at the previous calculation. α is learning ratio, and β is momentum term that is used to adjust weights. In this paper, $\alpha = 0.65$ and $\beta = 0.9$ are used.

(6) Square error, occurred in one cycle, can be found by (18) as

$$e = \sum 0.5 \left|t_k - o_k\right|^2. \tag{18}$$

The completion of training the BPN, relative error (RE) for each data, and mean relative error (MRE) for all data are calculated according to (19), respectively, as

$$\begin{aligned} \text{RE} &= \left(\frac{100\left(t_k - o_k\right)}{t_k}\right) \\ \text{MRE} &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{100\left(t_k - o_k\right)}{t_k}\right), \end{aligned} \tag{19}$$

where n is the number of data [14, 15].

Table 1: System specifications.

$m_s = 243\,\text{kg}$	$k_s = 14671\,\text{N/m}$
$m_u = 40\,\text{kg}$	$k_t = 124660\,\text{N/m}$
$b_s = 370\,\text{N/(m/s)}$	$A_P = 3.35 \times 10^{-4}\,\text{m}^2$
$b_t = 414\,\text{N/(m/s)}$	$P_L = 10342500\,\text{Pa}$

4. Network Development

4.1. Input and Output Data. The speed of automotive is changed between 10 and 55 m/sec. The overshoot, settling time, and steady state error of system response are assumed between 1% and 10%, 0.3 and 1.5 second, and 0% and 2%, respectively. These parameters are the input value of network. Finally, the outputs of net are the gain parameters of the PID controller.

4.2. Network Configuration. The nodes at the input and output layer are determined by the number of predictor and predicted variables. In this research, there are 4 nodes in the input layers due to the number of input variables, and 3 nodes in the output layer, for similar reasons. There are no rules given to determine the exact number of hidden layers and the number of nodes in hidden layers. A large number of hidden-layer nodes will lead to an overfit at intermediate points, which can slow down the operation of NN. On the other hand, an accurate output may not be achieved if too few hidden layer nodes are included in the neural network. The results show that the best configuration of the network is achieved by one hidden layer. The number of nodes in the input layer, in the hidden layer, and in the output layer is chosen to 4–10–3, respectively. The activation function in the input and the hidden layers is sigmoid function and linear function in the output layer.

4.3. Preprocessing the Data. For a proper working of the neural network, a preprocessing of the input and output data is performed. The input values are normalized between −1 and 1, since the activation function is a sigmoid function in the input layer. Normalization is made by the following function:

$$x_{\text{norm}} = 2 \cdot \frac{x - x_{\min}}{x_{\max} - x_{\min}} - 1. \tag{20}$$

The output values are normalized between 0 and 1 and a linear function in the output layer [16].

4.4. Training of the Network. Once a network is structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. During the training, the weights are iteratively adjusted to minimize the network performance function. As performance function the mean square error, the average squared error between the network output and the target output is applied. For the training of the network the MATLAB Neural Network Toolbox is used [17]. The Levenberg-Marquardt algorithm is chosen to perform the training with the default values suggested in [18]. In this work, for training is used of three functions, newelm, newff, and newcf. The stopping criteria are adjusted; that is the mean square error should be less than 10^{-5} and the number of epochs (iterations) should be less than 5000. The BPN learning process involves a forward propagation pass calculating the outputs using the inputs, weights, and neuron transfer functions, as well as a back propagation pass correcting the weights using the error between the predicted and target values. The major advantage of the BPN model is its ability to learn from examples without requiring principal knowledge of domain problems. In addition, it is very effective in dealing with large amounts of data. The structure of the BPN model can easily be constructed according to the domain problem and the availability of data attributes.

Table 2: The test data values set used in the BPN.

Number	Overshoot (%)	Settling time (sec)	Steady state error (%)	Velocity (m/s)
1	1	0.35	1	10
2	2	0.44	2	15
3	0.5	0.52	1	20
4	1.5	1	0.09	25
5	4	1.2	0.085	30
6	5.5	1.3	2.5	35
7	6	1.4	0.05	40
8	7	1.5	0.35	45
9	2.5	0.65	0.15	50
10	10	0.73	0.1	55
11	3.5	0.84	0.01	52
12	4.3	1.25	0.012	32
13	1.52	0.55	0.001	23
14	4.5	1.27	0	42

Table 3: The variable training methods.

Acronym	Description
LM	Levenberg-Marquardt
BFG	BFGS quasinewton
RP	Resilient back propagation
SCG	Scaled conjugate gradient
CGB	Conjugate gradient with Powell/Beale restarts
CGF	Fletcher-Powell conjugate gradient
CGP	Polak-Ribiére conjugate gradient
OSS	One-step secant
GDX	Variable learning rate back propagation

Table 4: The variable activation functions in the layers.

Number	The activation function	
	Hidden layer	Output layer
1	logsig	logsig
2	logsig	tansig
3	logsig	purelin
4	tansig	tansig
5	tansig	logsig
6	tansig	purelin

Table 5: The (R^2) values for PID gain parameters with various neurons in the hidden layer.

Number of hidden neuron	Acronym of training method								
	LM	BFG	RP	SCG	CGB	CGF	CGP	OSS	GDX
7	0.9977	0.9888	0.9819	0.9957	0.9738	0.9918	0.9905	0.9913	0.9891
8	0.9981	0.9676	0.992	0.9909	0.9942	0.9864	0.9846	0.9969	0.9916
10	0.9999	0.9981	0.9942	0.9578	0.9986	0.9978	0.9681	0.9963	0.9945
15	0.9985	0.9998	0.9967	0.9996	0.9996	0.9995	0.9595	0.9890	0.9856
18	0.9967	0.9928	0.9994	0.9998	0.9998	0.9698	0.9898	0.9898	0.9881
20	0.9998	0.9499	0.999	0.9699	0.9997	0.9698	0.9699	0.9797	0.9950
25	0.9996	0.9699	0.9994	0.9699	0.9699	0.9599	0.9899	0.9479	0.9983
26	0.9944	0.9916	0.9996	0.9797	0.9694	0.9792	0.9799	0.9688	0.9782
27	0.9989	0.9599	0.9995	0.9899	0.9899	0.9899	0.9799	0.9896	0.9870
28	0.9998	0.9939	0.9994	0.9799	0.9999	0.9599	0.9799	0.9889	0.9984
29	0.9997	0.9419	0.9997	0.9799	0.9599	0.9699	0.9799	0.9889	0.9985

Table 6: The results of the variable training methods in the BPN with *newelm* function.

Acronym	Epoch in goal	Error goal	Train time (s)	Test time (s)
LM	153	met	28.724082	0.054526
BFG	1528	met	77.482451	0.046583
RP	3000	Not met	67.677318	0.046466
SCG	3000	Not met	101.328176	0.044047
CGB	2932	Not met	118.102075	0.052230
CGF	2125	Not met	88.894782	0.046125
CGP	2302	Not met	94.665335	0.046852
OSS	3000	Not met	112.071065	0.046677
GDX	3000	Not met	61.243551	0.046046

Table 7: The results of the variable training methods in the BPN with *newcf* function.

Acronym	Epoch in goal	Error goal	Train time (s)	Test time (s)
LM	251	met	38.726182	0.057241
BFG	1018	met	97.481454	0.042678
RP	2439	Not met	49.16278	0.048146
SCG	1980	Not met	33.19076	0.042047
CGB	3232	Not met	128.15905	0.055130
CGF	3025	Not met	108.15282	0.056985
CGP	2100	Not met	67.150335	0.044152
OSS	1000	met	59.011265	0.058677
GDX	3230	Not met	67.120551	0.056046

5. Numerical Results

Specifications of the suspension system used for simulation are given in Table 1. The control system is simulated subject to a road displacement shown by (8).

Table 8: The results of the variable training methods in the BPN with *newff* function.

Acronym	Epoch in goal	Error goal	Train time (s)	Test time (s)
LM	293	met	40.145082	0.054789
BFG	528	met	88.102451	0.056673
RP	1210	met	100.677318	0.056906
SCG	4000	Not met	97.949761	0.065237
CGB	3745	Not met	138.78905	0.062230
CGF	3450	Not met	112.89642	0.053565
CGP	3162	Not met	100.13335	0.056907
OSS	3400	Not met	160.14895	0.066677
GDX	2000	Not met	43.446511	0.056044

In this study, the back propagation learning algorithm is used in a feed forward, single hidden layer network. A variable transfer function is used as the activation function for both the hidden layer and the output layer. Many back propagation training algorithms were repeatedly applied until satisfactory training was achieved. The number of test data value used in the BPN is shown in Table 2. The names of training algorithms are shown in Table 3. The activation function for the hidden layer and the output layer that is used are shown in Table 4.

The best combination for all methods that is used in this paper is logsig for hidden layer and purelin for output layer. In the hidden layer, a number of neurons from 7 to 29 are used. The data set available for K_P, K_I, and K_D included 100 data patterns. K_P is called the proportional gain, K_I the integral gain, and K_D the derivative gain of a PID controller. From these, 80 data patterns were used for training the network, and the remaining 20 patterns were randomly selected and used as the test data set. The regression value (R^2) of the output variable values for the test data set for various neurons in hidden layer is shown in Table 5. It should be noted that these data were completely unknown to the network.

Table 9: Comparison between actual gain parameters of PID and the BPN model (with *newelm* function).

Number	K_P			K_I			K_D		
	Actual	Predicted	Percentage of error	Actual	Predicted	Percentage of error	Actual	Predicted	Percentage of error
1	12614.31	13017.95	3.25	16743.32	16949.26	1.23	28345.20	30091.26	6.16
2	14590.41	15830.59	8.50	15234.90	16011.87	5.10	27670.82	28365.35	2.51
3	12090.59	12634.66	4.50	16045.61	16474.02	2.67	26674.96	27013.73	1.27
4	14278.66	14649.90	2.60	16396.63	16955.75	3.41	27869.42	28752.88	3.17
5	14797.23	15674.70	5.93	16853.16	17341.83	2.90	27001.56	27322.87	1.19
6	14856.34	15683.83	5.57	16166.90	16186.30	0.12	25438.94	25487.28	0.17
7	14879.61	15821.48	6.33	15879.61	15965.35	0.54	26730.38	27075.20	1.29
8	14898.49	14916.36	0.12	15990.53	16204.80	1.34	28593.77	30134.97	5.39
9	14478.19	15394.65	6.33	15069.52	15841.07	5.12	27433.71	28720.35	4.69
10	13967.83	14526.54	4.00	16649.37	16919.08	1.62	28082.82	28695.02	2.18
11	12889.63	13119.06	1.78	16298.48	16818.40	3.19	29782.51	29830.16	0.16
12	12794.88	13164.65	2.89	15589.92	15903.27	2.01	25967.69	25970.28	0.01
13	12797.34	12799.25	0.015	15590.29	16446.19	5.49	27420.55	28026.54	2.21
14	13312.90	14244.80	7.00	15690.33	16189.28	3.18	28890.69	29269.15	1.31

Table 10: Comparison between actual gain parameters of PID and the BPN model (with *newcf* function).

Number	K_P			K_I			K_D		
	Actual	Predicted	Percentage of error	Actual	Predicted	Percentage of error	Actual	Predicted	Percentage of error
1	12614.31	12811.09	1.56	16743.32	18136.36	8.32	28345.20	29921.19	5.56
2	14590.41	14996.02	2.78	15234.90	15496.94	1.72	27670.82	29635.44	7.10
3	12090.59	12562.12	3.90	16045.61	17080.55	6.45	26674.96	27019.06	1.29
4	14278.66	14305.78	0.19	16396.63	16832.78	2.66	27869.42	29062.23	4.28
5	14797.23	15872.98	7.27	16853.16	17139.66	1.70	27001.56	29674.71	9.90
6	14856.34	15869.54	6.82	16166.90	16186.30	0.12	25438.94	26995.80	6.12
7	14879.61	15498.60	4.16	15879.61	15881.19	0.01	26730.38	28144.41	5.29
8	14898.49	15090.68	1.29	15990.53	17563.99	9.84	28593.77	28633.80	0.14
9	14478.19	15029.80	3.81	15069.52	16107.80	6.89	27433.71	29831.41	8.74
10	13967.83	14997.25	7.37	16649.37	17528.45	5.28	28082.82	31056.79	10.59
11	12889.63	13584.38	5.39	16298.48	17493.15	7.33	29782.51	32376.56	8.71
12	12794.88	13714.83	7.19	15589.92	16480.10	5.71	25967.69	26996.01	3.96
13	12797.34	14129.54	10.41	15590.29	16605.21	6.51	27420.55	28363.81	3.44
14	13312.90	14621.55	9.83	15690.33	16237.92	3.49	28890.69	30566.35	5.80

The closer this value is to unity, the better is the prediction accuracy. The best (R^2) value obtained is 0.9999, and it is obtained from the LM algorithm by 10 neurons in hidden layer.

In Tables 6, 7, and 8, the results of training the network using nine different training algorithms by 10 neurons in the hidden layer and logsig-purlin activation function are summarized. Each entry in the table represents 14 different trials, where different random initial weights are used in each trial.

The fastest algorithm for this problem is the LM. On the average, it is over two times faster than the next fastest algorithm. This is the type of problem for which the LM algorithm is best suited.

In Tables 9, 10, and 11 a comparison between the actual gain parameters of PID controller and prediction with the artificial neural network for the LM method is presented. The actual valves are obtained by the written code by MATLAB. As can be seen, the error with newelm function is very small.

Controlled and uncontrolled of sprung mass of vehicle is compared in displacement and acceleration as shown from Figures 4 and 5, respectively. Note that in these figures dash and solid lines are uncontrolled and controlled cases of sprung mass, respectively. The body displacement of controlled system is very smooth with maximum values of 0.0015 m and 0.0043 m while the uncontrolled system provides high oscillations with maximum values of 0.078 m and 0.1578 m. The passenger comfort is provided by controlling

Table 11: Comparison between actual gain parameters of PID and the BPN model (with *newff* function).

Number	K_P			K_I			K_D		
	Actual	Predicted	Percentage of error	Actual	Predicted	Percentage of error	Actual	Predicted	Percentage of error
1	12614.31	12775.77	1.28	16743.32	16818.66	0.45	28345.20	28827.06	1.70
2	14590.41	16301.86	11.73	15234.90	15516.74	1.85	27670.82	29992.40	8.39
3	12090.59	13617.63	12.63	16045.61	16457.98	2.57	26674.96	29507.84	10.62
4	14278.66	14685.60	2.85	16396.63	17701.80	7.96	27869.42	31205.38	11.97
5	14797.23	15801.96	6.79	16853.16	17773.34	5.46	27001.56	31483.81	16.60
6	14856.34	16102.78	8.39	16166.90	18824.73	16.44	25438.94	27245.10	7.10
7	14879.61	15251.60	2.50	15879.61	17230.96	8.51	26730.38	27521.59	2.96
8	14898.49	15485.49	3.94	15990.53	17490.44	9.38	28593.77	29820.44	4.29
9	14478.19	15242.63	5.28	15069.52	16379.06	8.69	27433.71	30064.60	9.59
10	13967.83	15138.33	8.38	16649.37	16922.41	1.64	28082.82	30127.24	7.28
11	12889.63	13516.06	4.86	16298.48	18495.51	13.48	29782.51	30300.72	1.74
12	12794.88	13740.42	7.39	15589.92	16004.61	2.66	25967.69	26557.15	2.27
13	12797.34	14010.52	9.48	15590.29	16329.26	4.74	27420.55	29200.14	6.49
14	13312.90	14645.52	10.01	15690.33	16459.15	4.90	28890.69	29147.81	0.89

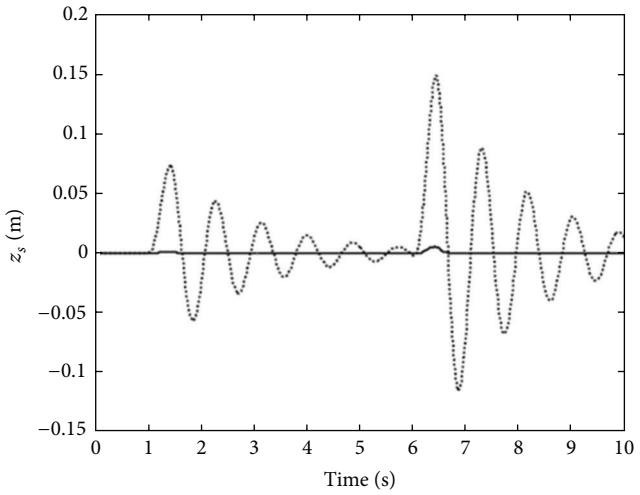

Figure 4: The body displacement.

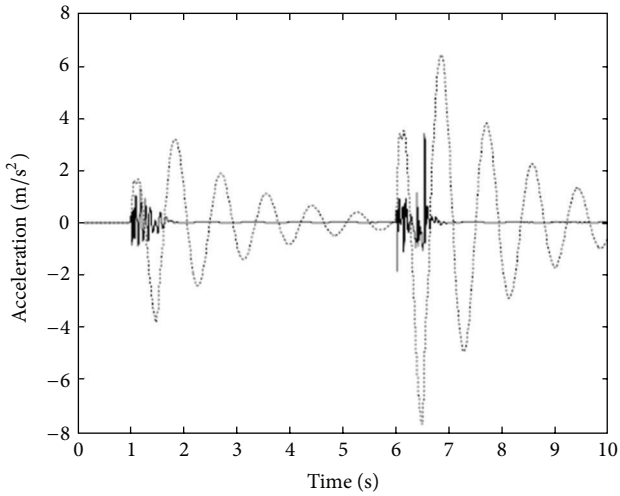

Figure 5: The body acceleration.

the body acceleration as shown in Figure 5. The controlled system reduces the acceleration successfully to zero after passing disturbances while uncontrolled case shows high accelerations.

6. Conclusions

The present study shows that for the analyses of PID controller of suspension system, the BPN is a suitable method. The BPN was successfully applied for determining the gain parameters of a PID controller for a suspension system. Date for developing the ANN model is obtained by the written code with MATLAB. Results from ANN model are compared with the results from the classical model. The best regression value for the simulation is 0.9999 with newelm function. The MRE value of the BPN model is 4.2%. The results show that newelm function is more accurate than newff and newcf functions. Also the Levenberg-Marquardt training is faster than other training methods. The BPN method also has the advantages of computational speed, low cost, and ease of use by people with little technical experience.

References

[1] H. Du and N. Zhang, "H_∞ control of active vehicle suspensions with actuator time delay," *Journal of Sound and Vibration*, vol. 301, no. 1-2, pp. 236–252, 2007.

[2] H. R. Karimi, "Optimal vibration control of vehicle engine-body system using Haar functions," *International Journal of Control, Automation and Systems*, vol. 4, no. 6, pp. 714–724, 2006.

[3] H. R. Karimi, M. Zapateiro, and N. Luo, "An LMI Approach to H_∞ control of vehicle engine-body vibration systems with time-varying actuator delay," *Proceedings of the Institution of Mechanical Engineers I*, vol. 222, pp. 883–894, 2008.

[4] A. Giua, C. Seatzu, and G. Usai, "A mixed suspension system for a half-car vehicle model," *Dynamics and Control*, vol. 10, no. 4, pp. 375–397, 2000.

[5] R. Guclu and K. Gulez, "Neural network control of seat vibrations of a non-linear full vehicle model using PMSM," *Mathematical and Computer Modelling*, vol. 47, no. 11-12, pp. 1356–1371, 2008.

[6] A. Hac, "Optimal linear preview control of active vehicle suspension," *Vehicle System Dynamics*, vol. 21, no. 3, pp. 167–195, 1992.

[7] S. Yildirim and I. Uzmay, "Statistical analysis of vehicles' vibration due to road roughness using radial basis artificial neural network," *Applied Artificial Intelligence*, vol. 15, no. 4, pp. 419–427, 2001.

[8] Ş. Yildirim and I. Uzmay, "Neural network applications to vehicle's vibration analysis," *Mechanism and Machine Theory*, vol. 38, no. 1, pp. 27–41, 2003.

[9] K. Spentzas and S. A. Kanarachos, "Design of a non-linear hybrid car suspension system using neural networks," *Mathematics and Computers in Simulation*, vol. 60, no. 3–5, pp. 369–378, 2002.

[10] C. Kim and P. I. Ro, "A sliding mode controller for vehicle active suspension systems with non-linearities," *Proceedings of the Institution of Mechanical Engineers D*, vol. 212, no. 2, pp. 79–91, 1998.

[11] S. Yildirim, "Vibration control of suspension systems using a proposed neural network," *Journal of Sound and Vibration*, vol. 277, no. 4-5, pp. 1059–1069, 2004.

[12] M. M. Fateh and S. S. Alavi, "Impedance control of an active suspension system," *Mechatronics*, vol. 19, no. 1, pp. 134–140, 2009.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[14] H. Zhang, W. Wu, and M. Yao, "Boundedness and convergence of batch back-propagation algorithm with penalty for feedforward neural networks," *Neurocomputing*, vol. 89, pp. 141–146, 2012.

[15] H. Shao and G. Zheng, "Convergence analysis of a back-propagation algorithm with adaptive momentum," *Neurocomputing*, vol. 74, no. 5, pp. 749–752, 2011.

[16] D. Gao, Y. Kinouchi, K. Ito, and Z. Zhao, "Neural networks for event extraction from time series: a back propagation algorithm approach," *Future Generation Computer Systems*, vol. 21, no. 7, pp. 1096–1105, 2005.

[17] H. Demuth and M. Beale, *Matlab Neural Networks Toolbox, User's Guide*, The Math Works, 1992–2001, http://www.mathWorks.com.

[18] D. Ballabio and M. Vasighi, "A Matlab toolbox for self organizing maps and supervised neural network learning strategies," *Chemometrics and Intelligent Laboratory Systems*, vol. 118, pp. 24–32, 2012.