Research Article

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

M. Heidari¹ and H. Homaei²

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

Copyright © 2013 M. Heidari and H. Homaei. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 $10^{-5}$. 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 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
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]:

\[
m_u \ddot{z}_u = -b_t (\dot{z}_u - \dot{z}_a) + k_t (z_u - z_a) - f_a
\]  

(1)

\[
m_s \ddot{z}_s = b_s (\dot{z}_s - \dot{z}_a) + k_s (z_s - z_a)
\]  

(2)

\[f_a = P_t A_p,
\]

(3)

where \(m_u\), \(m_s\), \(k_s\), \(k_t\), \(b_t\), and \(b_s\) 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 \(m_u\). \(P_t\) 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,

\[f_{\text{dis}} = b_t (\dot{z}_u - \dot{z}_a) + k_t (z_r - z_u),
\]

(4)

where \(f_{\text{dis}}\) is an applied force on the tire from the road. Equation (2) can be rewritten as

\[m_s \ddot{z}_u = b_s (\dot{z}_u - \dot{z}_a) + k_s (z_r - z_u) + f_{\text{dis}} - f_a.
\]

(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
When the control input \( f_a \) only was considered, \( z_r \) was set to zero. Thus, the transfer function \( G_1(s) \) can be written as follows:

\[
G_1(s) = \frac{Z_t(s) - Z_u(s)}{Z_r(s)} = \frac{(-m_b s^3 - m_t s^2)}{\Delta}
\]

\[
\Delta = \det \begin{bmatrix} m_s s^3 + b_s + k_t & -b_s + k_i \\ -b_s + k_t & (m_s s^3 + b_s + b_t) s + (k_t + k_i) \end{bmatrix}.
\]

(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:

\[
G_2(s) = \frac{Z_t(s) - Z_u(s)}{f_a(s)} = \frac{(m_t + m_u) s^3 + b_t s + k_t}{\Delta}.
\]

(7)

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

\[
z_r(t) = (0.01m) \sin(\omega_b t),
\]

\[
\omega_b = 0.2909v,
\]

(8)

(9)

where \( v \) is velocity of car on km/h, \( \omega_b \) 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

\[
f(t) = K_P e(t) + K_I \int e(t) dt + K_D \frac{de(t)}{dt}.
\]

(10)

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

\[
e(t) = y_d(t) - y(t),
\]

(11)

where \( y_d(t) \) is the desired car amplitude of displacement and \( y(t) \) is the current measured car amplitude. \( K_P \) is 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.

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))

\[ y = f \left( \sum_{i} u_i \right), \quad (12) \]

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

\[ \text{net}_i = \sum x_i w_i \]

\[ y_i = f \left( \text{net}_i \right), \quad (13) \]

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

\[ \text{net}_k = \sum y_i w_{jk} \]

\[ o_k = f \left( \text{net}_k \right), \quad (14) \]

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

\[ f \left( \text{net}_i \right) = \frac{1}{1 + e^{-\text{net}_i}}, \quad \text{logsig} \]

\[ f \left( \text{net}_i \right) = \frac{1 - e^{-\text{net}_i}}{1 + e^{-\text{net}_i}}, \quad \text{tansig} \quad (15) \]

\[ f \left( \text{net}_i \right) = \text{net}_i, \quad \text{linear}. \]

4. Errors made at the end of one cycle are

\[ e_k = (t_k - o_k) o_k (1 - o_k) \]

\[ e_i = y_i (1 - y_i) \sum e_k w_{ij}, \quad (16) \]

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

\[ w_{jk} = w_{jk} + \alpha e_k y_i + \beta \Delta w_{jk} \]

\[ w_{ij} = w_{ij} + \alpha e_i x_i + \beta \Delta w_{ij}, \quad (17) \]

where \( w_{ij} \) and \( \Delta w_{ij} \) are weights of output layer. \( \Delta w_{jk} \) and \( \Delta w_{ij} \) are correction made in weights at the previous calculation. \( \alpha \) is learning ratio, and \( \beta \) 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

\[ e = \sum 0.5 | t_k - o_k |^2. \quad (18) \]

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

\[ \text{RE} = \left( \frac{100 \left( t_k - o_k \right)}{t_k} \right) \]

\[ \text{MRE} = \frac{1}{n} \sum_{k=1}^{n} \left( \frac{100 \left( t_k - o_k \right)}{t_k} \right), \quad (19) \]

where \( n \) is the number of data [14, 15].
**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 layer 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:

\[ x_{\text{norm}} = 2 \cdot \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} - 1. \]  

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

<table>
<thead>
<tr>
<th>Number of hidden neuron</th>
<th>Acronym of training method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LM</td>
</tr>
<tr>
<td>7</td>
<td>0.9977</td>
</tr>
<tr>
<td>8</td>
<td>0.9981</td>
</tr>
<tr>
<td>10</td>
<td>0.9999</td>
</tr>
<tr>
<td>15</td>
<td>0.9967</td>
</tr>
<tr>
<td>20</td>
<td>0.9998</td>
</tr>
<tr>
<td>25</td>
<td>0.9996</td>
</tr>
<tr>
<td>26</td>
<td>0.9944</td>
</tr>
<tr>
<td>27</td>
<td>0.9989</td>
</tr>
<tr>
<td>28</td>
<td>0.9998</td>
</tr>
<tr>
<td>29</td>
<td>0.9997</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Epoch in goal</th>
<th>Error goal</th>
<th>Train time (s)</th>
<th>Test time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM</td>
<td>251</td>
<td>met</td>
<td>38.726182</td>
<td>0.054789</td>
</tr>
<tr>
<td>BFG</td>
<td>528</td>
<td>met</td>
<td>88.102451</td>
<td>0.056673</td>
</tr>
<tr>
<td>RP</td>
<td>1210</td>
<td>met</td>
<td>100.677318</td>
<td>0.056906</td>
</tr>
<tr>
<td>SCG</td>
<td>4000</td>
<td>Not met</td>
<td>97.949761</td>
<td>0.065237</td>
</tr>
<tr>
<td>CGB</td>
<td>3745</td>
<td>Not met</td>
<td>138.78905</td>
<td>0.062230</td>
</tr>
<tr>
<td>CGF</td>
<td>3450</td>
<td>Not met</td>
<td>112.89642</td>
<td>0.053565</td>
</tr>
<tr>
<td>CGP</td>
<td>3162</td>
<td>Not met</td>
<td>101.13335</td>
<td>0.056907</td>
</tr>
<tr>
<td>OSS</td>
<td>3400</td>
<td>Not met</td>
<td>160.14895</td>
<td>0.066677</td>
</tr>
<tr>
<td>GDX</td>
<td>2000</td>
<td>Not met</td>
<td>43.446511</td>
<td>0.056044</td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>Number</th>
<th>Actual</th>
<th>Predicted</th>
<th>Percentage of error</th>
<th>Actual</th>
<th>Predicted</th>
<th>Percentage of error</th>
<th>Actual</th>
<th>Predicted</th>
<th>Percentage of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12614.31</td>
<td>13017.95</td>
<td>3.25</td>
<td>16743.32</td>
<td>16494.26</td>
<td>1.23</td>
<td>28345.20</td>
<td>30091.26</td>
<td>6.16</td>
</tr>
<tr>
<td>2</td>
<td>14590.41</td>
<td>15380.59</td>
<td>5.80</td>
<td>15234.90</td>
<td>16011.87</td>
<td>5.10</td>
<td>27670.82</td>
<td>28365.35</td>
<td>2.51</td>
</tr>
<tr>
<td>3</td>
<td>12090.59</td>
<td>12634.66</td>
<td>4.50</td>
<td>16045.61</td>
<td>16474.02</td>
<td>2.67</td>
<td>26674.96</td>
<td>27013.73</td>
<td>1.27</td>
</tr>
<tr>
<td>4</td>
<td>14278.66</td>
<td>14649.90</td>
<td>2.60</td>
<td>16396.63</td>
<td>16955.75</td>
<td>3.41</td>
<td>27869.42</td>
<td>28752.88</td>
<td>3.17</td>
</tr>
<tr>
<td>5</td>
<td>14797.23</td>
<td>15674.70</td>
<td>5.93</td>
<td>16853.16</td>
<td>17341.83</td>
<td>2.90</td>
<td>27001.56</td>
<td>27322.87</td>
<td>1.19</td>
</tr>
<tr>
<td>6</td>
<td>14856.34</td>
<td>15683.83</td>
<td>5.57</td>
<td>16166.90</td>
<td>16818.30</td>
<td>0.12</td>
<td>25438.94</td>
<td>25487.28</td>
<td>0.17</td>
</tr>
<tr>
<td>7</td>
<td>14879.61</td>
<td>15821.48</td>
<td>6.33</td>
<td>15879.61</td>
<td>15965.35</td>
<td>0.54</td>
<td>26730.38</td>
<td>27075.20</td>
<td>1.29</td>
</tr>
<tr>
<td>8</td>
<td>14898.49</td>
<td>14916.36</td>
<td>0.12</td>
<td>15990.53</td>
<td>16204.80</td>
<td>1.34</td>
<td>28593.77</td>
<td>30134.97</td>
<td>5.39</td>
</tr>
<tr>
<td>9</td>
<td>14478.19</td>
<td>15394.65</td>
<td>6.33</td>
<td>15069.52</td>
<td>15841.07</td>
<td>5.12</td>
<td>27433.71</td>
<td>28720.35</td>
<td>4.69</td>
</tr>
<tr>
<td>10</td>
<td>13967.83</td>
<td>14526.54</td>
<td>4.00</td>
<td>16045.61</td>
<td>16474.02</td>
<td>2.67</td>
<td>26674.96</td>
<td>27013.73</td>
<td>1.27</td>
</tr>
<tr>
<td>11</td>
<td>12889.63</td>
<td>13119.06</td>
<td>1.78</td>
<td>16298.48</td>
<td>16818.40</td>
<td>3.19</td>
<td>29782.51</td>
<td>29830.16</td>
<td>0.16</td>
</tr>
<tr>
<td>12</td>
<td>12794.88</td>
<td>13164.65</td>
<td>2.89</td>
<td>15879.61</td>
<td>15903.27</td>
<td>0.54</td>
<td>26730.38</td>
<td>27075.20</td>
<td>1.29</td>
</tr>
<tr>
<td>13</td>
<td>12797.34</td>
<td>12799.25</td>
<td>0.01</td>
<td>15590.29</td>
<td>16446.19</td>
<td>5.49</td>
<td>27420.55</td>
<td>28026.54</td>
<td>2.21</td>
</tr>
<tr>
<td>14</td>
<td>13312.90</td>
<td>14244.80</td>
<td>7.00</td>
<td>15690.33</td>
<td>16189.28</td>
<td>3.18</td>
<td>28890.69</td>
<td>29269.15</td>
<td>1.31</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Number</th>
<th>Actual $K_P$</th>
<th>Predicted $K_P$</th>
<th>Percentage of error</th>
<th>Actual $K_I$</th>
<th>Predicted $K_I$</th>
<th>Percentage of error</th>
<th>Actual $K_D$</th>
<th>Predicted $K_D$</th>
<th>Percentage of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12614.31</td>
<td>12775.77</td>
<td>1.28</td>
<td>16743.32</td>
<td>16818.66</td>
<td>0.45</td>
<td>28345.20</td>
<td>28827.06</td>
<td>1.70</td>
</tr>
<tr>
<td>2</td>
<td>14590.41</td>
<td>16301.86</td>
<td>11.73</td>
<td>15234.90</td>
<td>15516.74</td>
<td>1.85</td>
<td>27670.82</td>
<td>29992.40</td>
<td>8.39</td>
</tr>
<tr>
<td>3</td>
<td>12090.59</td>
<td>13617.63</td>
<td>12.63</td>
<td>16045.61</td>
<td>16457.98</td>
<td>2.57</td>
<td>26674.96</td>
<td>29507.84</td>
<td>10.62</td>
</tr>
<tr>
<td>4</td>
<td>14278.66</td>
<td>14685.60</td>
<td>2.85</td>
<td>16396.63</td>
<td>17701.80</td>
<td>7.96</td>
<td>27869.42</td>
<td>31205.38</td>
<td>11.97</td>
</tr>
<tr>
<td>5</td>
<td>14797.23</td>
<td>15801.96</td>
<td>6.79</td>
<td>16853.16</td>
<td>17773.34</td>
<td>5.46</td>
<td>27001.56</td>
<td>31483.81</td>
<td>16.60</td>
</tr>
<tr>
<td>6</td>
<td>14856.34</td>
<td>16027.78</td>
<td>8.39</td>
<td>16166.90</td>
<td>18824.73</td>
<td>16.44</td>
<td>25438.94</td>
<td>27245.10</td>
<td>7.10</td>
</tr>
<tr>
<td>7</td>
<td>14879.61</td>
<td>15251.60</td>
<td>2.50</td>
<td>15879.61</td>
<td>17230.96</td>
<td>8.51</td>
<td>26730.38</td>
<td>27521.59</td>
<td>2.96</td>
</tr>
<tr>
<td>8</td>
<td>14898.49</td>
<td>15485.49</td>
<td>3.94</td>
<td>15990.53</td>
<td>17490.44</td>
<td>9.38</td>
<td>28593.77</td>
<td>29820.44</td>
<td>4.29</td>
</tr>
<tr>
<td>9</td>
<td>14478.19</td>
<td>15242.63</td>
<td>5.28</td>
<td>15069.52</td>
<td>16379.06</td>
<td>8.69</td>
<td>27433.71</td>
<td>30064.60</td>
<td>9.59</td>
</tr>
<tr>
<td>10</td>
<td>13967.83</td>
<td>14010.52</td>
<td>8.38</td>
<td>16649.37</td>
<td>19224.41</td>
<td>1.64</td>
<td>28082.82</td>
<td>30127.24</td>
<td>7.28</td>
</tr>
<tr>
<td>11</td>
<td>12889.63</td>
<td>13516.06</td>
<td>4.86</td>
<td>16298.48</td>
<td>18495.51</td>
<td>13.48</td>
<td>29782.51</td>
<td>30300.72</td>
<td>1.74</td>
</tr>
<tr>
<td>12</td>
<td>12794.88</td>
<td>13740.42</td>
<td>7.39</td>
<td>15589.92</td>
<td>16004.61</td>
<td>2.66</td>
<td>25967.69</td>
<td>26557.15</td>
<td>2.27</td>
</tr>
<tr>
<td>13</td>
<td>12797.34</td>
<td>14010.52</td>
<td>9.48</td>
<td>15590.29</td>
<td>16329.26</td>
<td>4.74</td>
<td>27420.55</td>
<td>29200.14</td>
<td>6.49</td>
</tr>
<tr>
<td>14</td>
<td>13312.90</td>
<td>14645.52</td>
<td>10.01</td>
<td>15690.33</td>
<td>16459.15</td>
<td>4.90</td>
<td>28890.69</td>
<td>29147.81</td>
<td>0.89</td>
</tr>
</tbody>
</table>

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 newcfc 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


Submit your manuscripts at http://www.hindawi.com