Engng Applic. Artif. Intell. Vol. 4, No. 1, pp. 59–64, 1991
Printed in Great Britain.

0952-1976/91 $3.00+0.00

Contributed Paper

Neural Network Simulation of an Automotive Shock Absorber

JOSEPH GIACOMIN
Centro Ricerche Fiat

This paper discusses the use of a back-propagation neural network for the purpose of modeling an automotive shock absorber. A brief description of the method is presented, as well as the structure of the network developed. The absorber model has been implemented in a numerical suspension simulation and comparisons are made between the simulation outputs and experimental test results.

Keywords: Shock absorber, neural network, nonlinear dynamics.

LIST OF SYMBOLS

err	error output
$f(t)$	reaction force
$x(t)$	displacement
$\dot{x}(t)$	velocity
$x_n(t)$	normalized displacement
$\dot{x}_n(t)$	normalized velocity
I	input to a processing unit
W	weight value
X	output from a processing unit

INTRODUCTION

In the automotive industry much use is made of dynamic vehicle simulations. One typical simulation of interest is that of the behavior of a suspension system when passing over an obstacle (Fig. 1). A suspension design can be studied and optimized for ride comfort by numerical testing with a simulation code and standardized obstacles.

One of the most difficult suspension elements to model acceptably for simulation purposes is the shock absorber. The reaction force generated by the absorber is a function of several variables, among which are its displacement, velocity and frequency of oscillation.[1] The behavior of the shock absorber has traditionally been specified by force–velocity curves which are produced by plotting the maximum force vs the maximum velocity measured during a test cycle on a stand which imparts a known motion (sine wave, triangular wave, etc.) to the absorber. This specification accurately describes the peak values attained but lacks information about the behavior at all other points of the cycle. When inserting this characteristic curve into a suspension simulation model large errors can be produced in the calculated output.

The necessity of improving the quality of suspension simulations requires that procedures be found to model the absorber accurately. The study presented here involves a neural network modeling of a typical passenger car shock absorber.

Correspondence should be sent to: J. Giacomin, Centro Ricerche Fiat, Strada Torino 50, 10043 Orbassano (TO), Italy.

BACK-PROPAGATION NETWORKS

Back-propagation networks are well-suited to many engineering applications.[2–4] These networks have the general structure shown in Fig. 2. As can be seen, the basic building blocks of this neural network are elements (activation functions) and connections (weights).

Elements

Elements serve the purpose of collecting and modifying data inputs. Each element can be thought of as processing a small piece of the overall information content. The strength of neural network techniques lies in the use in parallel of many elements to simulate a system.

Each element sums the input from the preceding elements to which it is connected and transforms it in some fixed way (the activation function) to form its output. A typical processing element is shown in Fig. 3. The input to this element is the sum of the activations of the preceding elements to which is it connected, multiplied by the weight value of each connection. This can be written as

$$I_j = \sum_{i=1}^{n} W_{ij} X_i \tag{1}$$

where: I_j = the input to element j. W_{ij} = the strength value of the connection from element i to element j. X_i = the output from element i (in a previous layer).

The output of element j is given by transforming the input I_j by an activation function. Any activation function can be used with the back-propagation algorithm, subject to the restriction that there exists the first derivative. A typical activation function is simply the

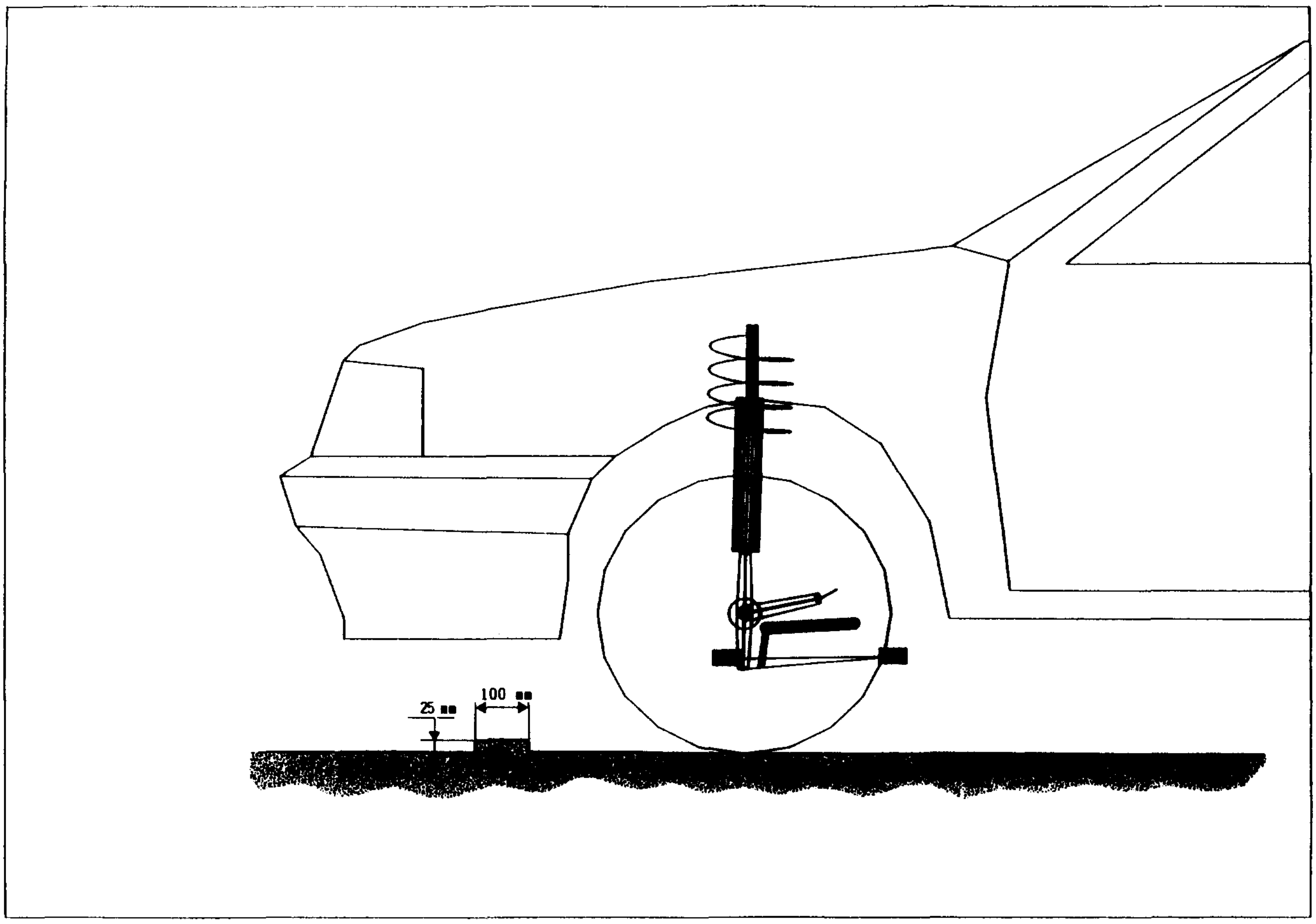

Fig. 1. McPherson front suspension model.

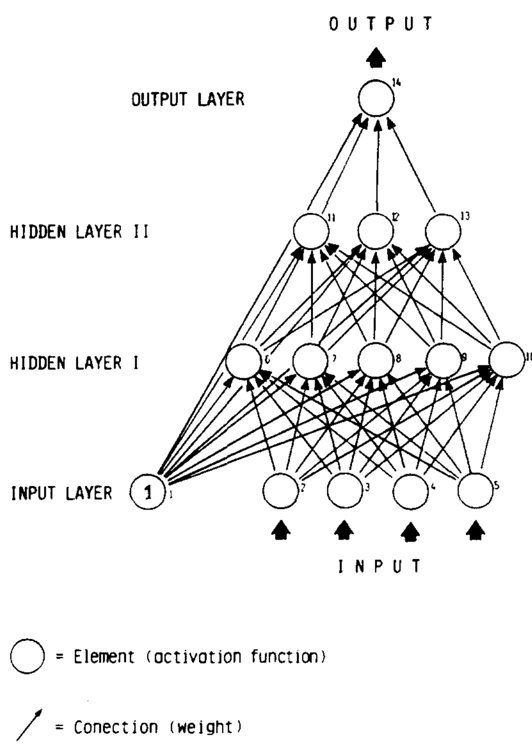

Fig. 2. Neural network.

linear transfer function which passes the input value straight through without modification to output. This can be written as:

$$X_j = I_j \tag{2}$$

In practice a network composed of only linear activation functions is limited as to the systems it can satisfactorily model. Among the various activation functions used to model highly nonlinear systems is the hyperbolic tangent function:

$$X_j = \frac{e^{I_j} - e^{-I_j}}{e^{I_j} + e^{-I_j}}. \tag{3}$$

This activation function is of interest because it exhibits a saturation behavior at both extremes which is typical of many biological mechanisms,[5] and is useful for our application because it introduces a strong nonlinearity.

Elements (activation functions) are static properties of the network, they are part of the basic structure. Once the elements of the network are all assigned an activation function, these functions remain in place throughout the various phases of use of the network.

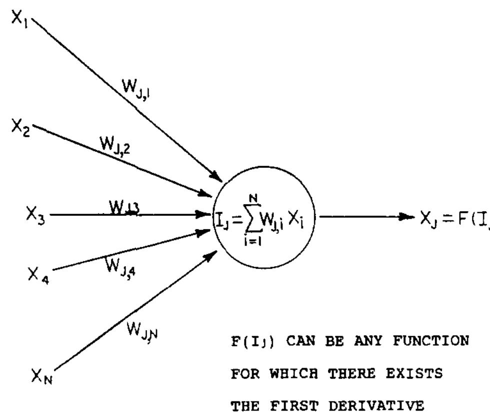

Fig. 3. Individual processing element.

Weights

Weights are the dynamic structure of a neural network that adjust to perform a required task. Each weight regulates the strength of the connection between two elements. Each element in the network receives as its input the output of the previous element to which it is attached, multiplied by the weight value on the connection between the two elements.

Organization

The elements and connections which form a back-propagation neural network are usually organized into layers. Figure 2 shows a network formed of an input layer, an output layer and several intermediate layers. The input layer has one element for each element of the input data pattern, while the output layer has one element for each of the desired outputs.

Functioning

In our study there are two distinct phases in the use of the back-propagation network: training and application. In the first phase the network is shown input/output pairs from the patterns or signals which the user wishes to simulate. A process involving a learning algorithm takes place, in which the connection weights of the network are adjusted with each set of input/output data so as to better reproduce the desired behavior. In the second phase the defined (trained) network is implemented through software to perform the desired task. In this phase the user can do away with the overhead involved in the training algorithm and keep only the simple weight and connection data.

The real conceptual and programming effort lies in the training of the network by the learning algorithm. What follows is a simplified description of the back-propagation learning algorithm.

The network initially begins with weight values connecting the various elements which are small (close to zero) random values. This is done because the learning algorithm would not be able to move if the start weight values were zero [see equations (1), (2) and (3)]. A set of values from the input data file is assigned to the input units. These values are propagated forward through the network by summing element inputs and calculating element outputs from the first layer to the last. At the output layer the calculated values are compared to the desired values. The difference is then propagated back toward the input units as an error signal. As this signal reaches each individual connection in the network the weight is modified so as to reduce the overall modeling error. The learning rule is a gradient descent algorithm which minimizes the overall system error. In its simplest form the function to perform this task can be written as:*

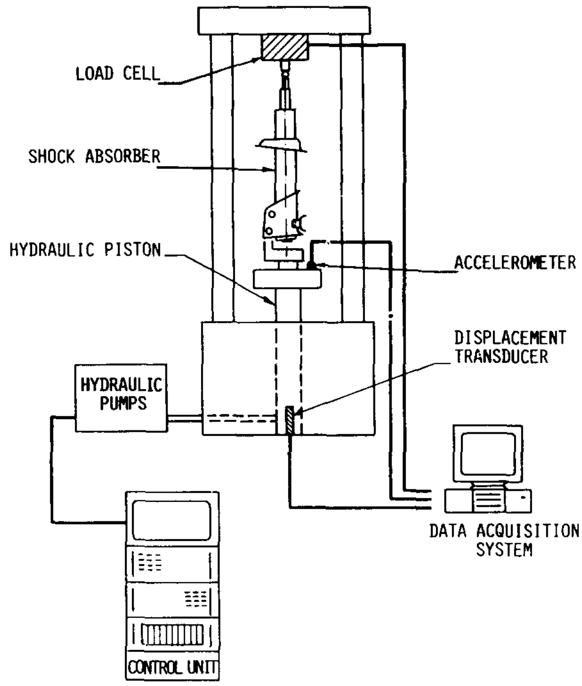

Fig. 4. Test setup.

$$\Delta W_{ij} = C\, err_j\, X_i. \tag{4}$$

where: ΔW_{ij} is the change to be made to the weight from element i to element j. C is a learning coefficient chosen according to the application (typically in the range 0.1–0.8). err_j is the error output from the element j. X_i is the activation value of element i.

The process of fixing input/output values and changing weights is repeated for a number of data sets. With each iteration the weights are modified such that the network more closely produces the correct output from the given input. The learning process is continued until the network converges to an optimal solution.

TRAINING DATA

The data used for training the neural network in this study was produced by testing a shock absorber in a test rig as shown in Fig. 4. In the test setup a known motion is imparted to the absorber by a hydraulic actuator and a PID control system.

For this study the shock absorber was considered to be a one degree of freedom oscillator with nonlinear

* For a complete derivation of this result see Ref. 6.

stiffness and damping terms. The quantities of interest were therefore the absorber displacement, velocity and reaction force. The quantities physically measured during the bench tests were the absorber displacement, acceleration and reaction force.

To generate a model suitable for obstacle studies it was decided to test the absorber with a series of sinusoids of frequencies from 1 to 30 Hz and amplitudes from 5 to 50 mm. Table 1 lists the various tests performed. The choice of tests was such as to cover a significant portion of the absorber's operating range. The tests were performed in few cycles (typically 6–8) in order to limit the heating of the absorber, and thus reduce the effects of temperature. The data selected for this study involved only the steady state portion of the test data, the transient having always died out in less than one test cycle.

Several operations were performed on the test data in order to prepare it for use as training data. The first operation involved a numerical integration and trend removal of the acceleration signal to produce a velocity signal. With this the state variables $(x, \dot{x})$ were defined for use as model inputs.

The second operation performed was a normalization of the three data vectors (displacement, velocity and reaction force) in order to be in the range from -1 to $+1$. Normalization of the input/output data was performed to improve the accuracy of computation.

The last operation involved the order of presentation of the training data. The input/output data groups composed of the terms $x_n(t-1)$, $\dot{x}_n(t-1)$, $x_n(t)$, $\dot{x}_n(t)$ and $f(t)$ were rearranged in random order in the input data file. This was done to reduce the effects of unmeasured parameters and noise which vary somewhat between the different test cycles performed. The randomization of the presentation order made it harder for the network to overtrain[7] to any one test cycle and thus also to its noise. This permitted greater flexibility in the choice of training parameters.

SHOCK ABSORBER MODEL

It was decided that the neural model would take the form of an input layer for distributing the data and a series of nonlinear intermediate layers for performing the modeling. Several different back-propagation networks were tried empirically in order to optimize the size of the resulting network with respect to the accuracy of the produced output.

The final network decided upon is shown in Fig. 2. This network has four normalized inputs which are the displacement and velocity at times t and $t-1$. Initial tests showed that the network needed a velocity lag term $\dot{x}(t-1)$ as well as $\dot{x}(t)$ to perform adequately. In this way the network had information on the slope of the velocity curve, hence the acceleration. These initial attempts also showed that the modeling accuracy was insensitive to additional lag terms. The signal output from the network is the normalised force generated at time t.

The activation functions decided upon were linear for the input and the output elements and hyperbolic tangent for the elements of the two intermediate layers. The choice of linear functions for the input layer was arbitrary. For the nonlinear layers both the sigmoid and hyperbolic tangent function were initially tried, and testing showed that both functions produced equally accurate results. The hyperbolic tangent function was eventually preferred because it changes sign (range -1 to 1) as in the test data (which was useful for debugging). The output activation function was chosen linear because initial tests showed convergence problems when the activation function of the output element was nonlinear.

There is an additional input element which is the bias term of constant value 1. This term creates an offset value to the inputs of the various elements of the network.

During training the network reached near-optimum performance after a few hundred cycles. With the weight data defined, the operation of this network can be summarized by the following steps:

(1) Normalize input data:

$$x_n(t-1) = x(t-1)/0.05$$
$$\dot{x}_n(t-1) = \dot{x}(t-1)/3.0$$
$$x_n(t) = x(t)/0.05$$
$$\dot{x}_n(t) = \dot{x}(t)/3.0$$

where $x(t-1)$, $\dot{x}(t-1)$, $x(t)$ and $\dot{x}(t)$ are to be in meters

Table 1. Shock absorber tests performed

	A(mm)										
F(Hz)	5	8	10	13	15	18	20	25	30	40	50
30	942.4	1507.9	1884.9	2450.3	2827.3						
25	785.4	1256.6	1570.8	2042.0	2356.2	2827.4	3141.6				
20	628.3	1005.3	1256.6	1633.6	1884.9	2261.9	2513.2	3141.6			
15	471.2	753.9	942.4	1225.2	1413.7	1696.4	1884.9	2356.1	2827.4		
10	314.6	502.6	628.3	816.8	942.2	1130.9	1256.6	1570.9	1884.9	2513.3	3141.6
7	219.9	351.8	439.8	571.7	659.7	791.6	879.6	1099.5	1319.5	1759.3	2199.1
5	157.0	251.3	314.1	408.4	471.2	565.4	628.3	785.4	942.4	1256.6	1570.8
3	94.2	150.8	188.4	245.0	282.7	339.9	376.9	471.2	565.5	753.9	942.4
1	31.4	50.2	62.8	81.8	94.2	113.0	125.6	157.0	188.4	251.3	314.1

The values given in the table indicate the maximum theoretical velocity attainable with that particular combination of frequency and amplitude.

and m/s respectively. The normalization values of 3.0 m/s and 0.05 m are the maximum values in the training data used.

(2) Calculate the inputs to the elements in the first hidden layer:

$$I_6 = -1.4077 - 1.6966x_n(t-1) + 1.6029\dot{x}_n(t-1) + 2.0009x_n(t) + 0.1834\dot{x}_n(t)$$

$$I_7 = -0.8718 + 0.6467x_n(t-1) - 2.0996\dot{x}_n(t-1) - 1.1381x_n(t) + 0.5000\dot{x}_n(t)$$

$$I_8 = -0.7522 - 3.3363x_n(t-1) + 2.5674\dot{x}_n(t-1) + 1.0525x_n(t) - 1.1593\dot{x}_n(t)$$

$$I_9 = 0.2825 + 1.8938x_n(t-1) - 2.4293\dot{x}_n(t-1) - 2.6375x_n(t) - 1.2123\dot{x}_n(t)$$

$$I_{10} = -1.2699 + 0.1900x_n(t-1) - 1.3099\dot{x}_n(t-1) - 3.3064x_n(t) + 3.4466\dot{x}_n(t).$$

(3) Calculate the inputs to the elements in the second hidden layer:

$$I_{11} = -0.0438 + 0.5584\,\mathrm{Tanh}(I_6) - 0.1036\,\mathrm{Tanh}(I_7) + 1.5582\,\mathrm{Tanh}(I_8) - 1.6051\,\mathrm{Tanh}(I_9) - 2.0515\,\mathrm{Tanh}(I_{10})$$

$$I_{12} = -0.5539 - 1.3102\,\mathrm{Tanh}(I_6) + 1.0968\,\mathrm{Tanh}(I_7) - 0.6469\,\mathrm{Tanh}(I_8) - 1.3891\,\mathrm{Tanh}(I_9) + 0.0067\,\mathrm{Tanh}(I_{10})$$

$$I_{13} = 0.2545 + 1.1748\,\mathrm{Tanh}(I_6) - 1.1324\,\mathrm{Tanh}(I_7) + 0.4965\,\mathrm{Tanh}(I_8) + 0.7793\,\mathrm{Tanh}(I_9) - 0.2556\,\mathrm{Tanh}(I_{10})$$

(4) Calculate the resultant output force:

$$f(t) = [0.0224 + 0.4646\,\mathrm{Tanh}(I_{11}) - 0.1943\,\mathrm{Tanh}(I_{12}) + 0.2004\,\mathrm{Tanh}(I_{13})]\,2038$$

where $f(t)$ is given in Newtons and the normalization value of 2038 is the maximum found in the reaction force data.

Figures 5(a) and (b) present two examples of the actual training force and the force calculated by the network given above. There is good agreement between the neural model and the actual absorber.

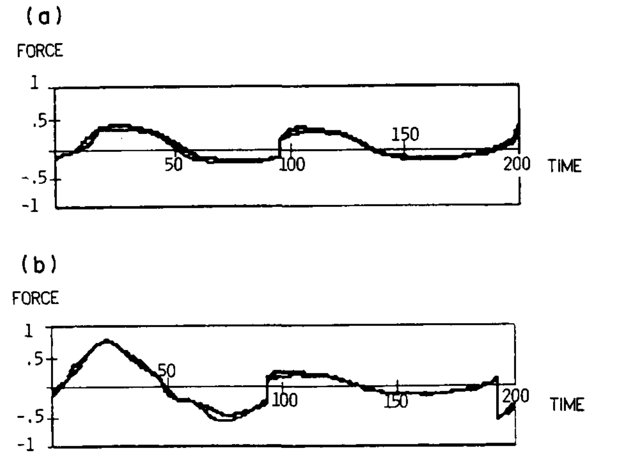

Fig. 5. Two examples of actual force and force calculated from the neural network.

SIMULATION RESULTS

Figures 6 and 7 present results obtained from simulations performed with the ADAMS[8] code and experimental tests in which a McPherson front-suspension model is run over a rectangular obstacle of width 100 mm and height 25 mm. The model includes the principal structural elements as well as several of the rubber elements in the suspension bushings. In Figs 6(a) and 7(a) the shock absorber model is the traditional force–velocity specification, in Figs 6(b) and 7(b) the model is the neural network. The force given in the figures is measured at a suspension attachment point. The neural model was found to significantly improve

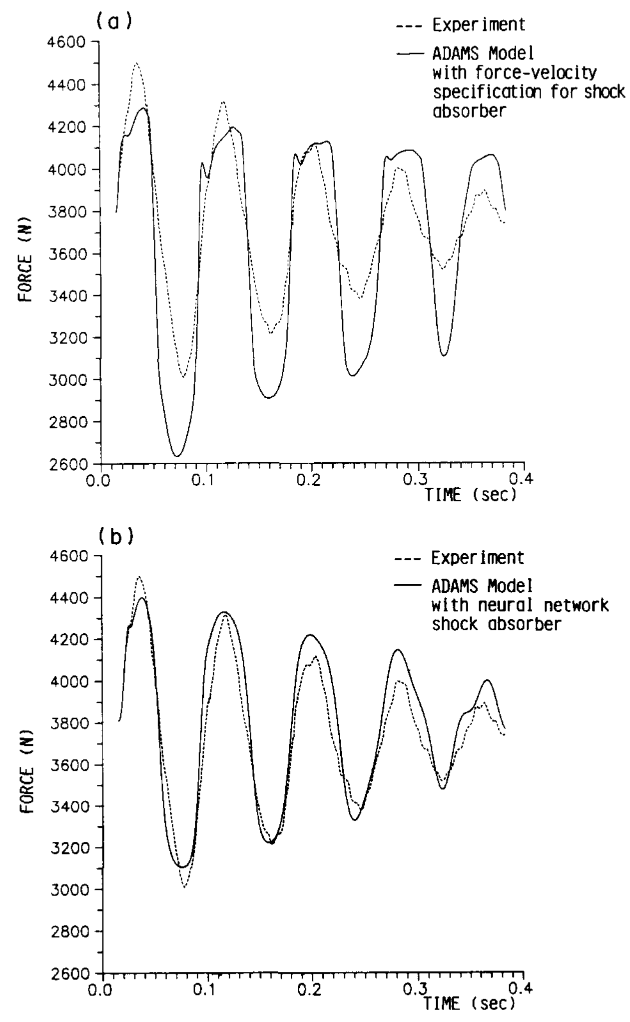

Fig. 6. Experimental and calculated forces at a suspension attachment point for passage over an obstacle of 100 by 25 mm at 40 km/h. (a) Force–velocity specification. (b) Neural network model.

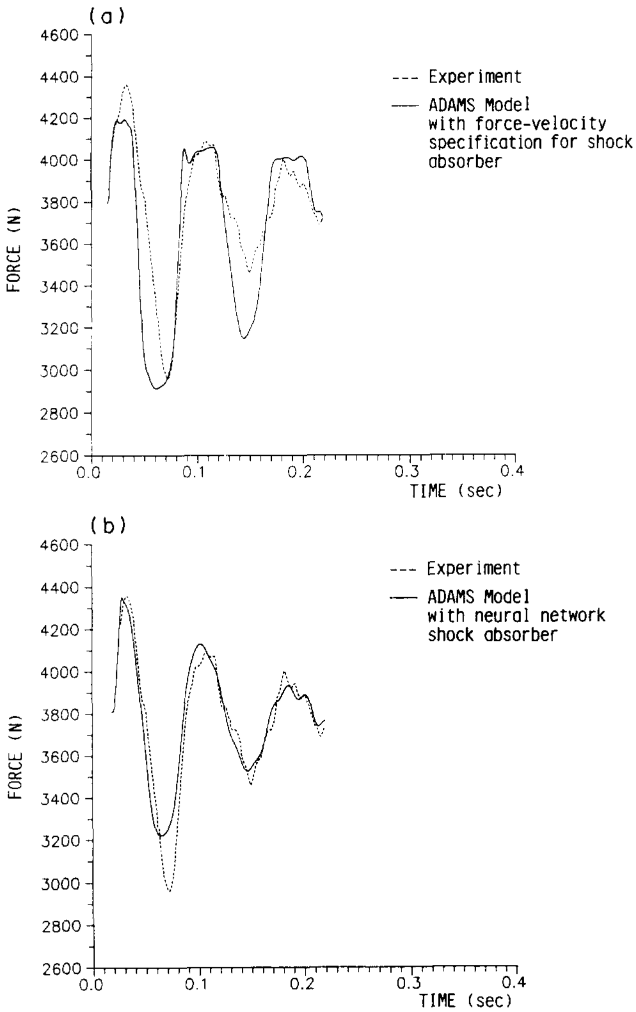

Fig. 7. Experimental and calculated forces at a suspension attachment point for passage over an obstacle of 100 by 25 mm at 80 km/h. (a) Force–velocity specification. (b) Neural network model.

the simulation results. The improvement was most evident for situations in which the shock absorber works near its operating limits.

CONCLUSIONS AND RECOMMENDATIONS

A back-propagation neural network was defined which effectively learned the behavior of an automotive shock absorber. In the simulations performed to date with the neural model there has been a substantial improvement in the results with respect to the traditional force—velocity specification.

The differences between the calculated and experimental results are to be attributed in part to the Adams suspension model and in part to the neural shock absorber model. The Adams modeling can best be improved by utilizing more-accurate shape and material property data for the various elements which compose the suspension.

The neural shock absorber model can also be improved in several ways. First, a more exhaustive set of training data (other vibratory waveforms) can be attempted. This may permit the neural network to see behavior which might not have been produced by the present training set (the absorber is nonlinear). A second improvement could be the utilization of more input parameters. For example, the reaction force has been shown to depend on the shock absorber oil temperature.[1] The fact that this parameter is not utilized here as an input to the model plays some part in the discrepancies of Figs 6 and 7.

Any steps towards determining improvements to the neural model must necessarily pass through verifications of the present model's behavior when subjected to other vibratory waveforms such as random vibration. The network in the current simulations has performed adequately when exposed to a motion significantly different from the test data set. Utilizing the model with waveforms which are greatly different in nature will bring to light any eventual deficiencies.

It is the object of current research to continue improving the neural model and to determine if this model or some derivative can perform adequately for all given input motions.

Acknowledgements—The author would like to express his thanks to Mr S. Girard for his help in testing the shock absorber, Mr M. Urbinati for his help in generating and using the Adams suspension model and Mr G. Burzio for his thought-stimulating discussions on the subject of neural networks.

REFERENCES

1. Lang H. A study of the characteristics of automotive hydraulic dampers at high stroking frequencies. Ph.D. Dissertation, The University of Michigan (1977).
2. Farhat N. and Baocheng B. Echo inversion and target shape estimation by neuromorphic processing. *Neural Networks*, **2**, 117–125 (1989).
3. Kuperstein M. and Wang J. Neural controller for adaptive movements with unforeseen payloads. *IEEE Trans. Neural Networks*, **1**, 137–142 (1990).
4. Rumelhart D. and McClelland J. *Explorations in Parallel Distributed Processing: a handbook of Models, Programs and Exercises*. MIT Press, Cambridge (1988).
5. Rumelhart D., McClelland J. and PDP Research Group *Parallel Distribution Processing: Explorations in the Microstructure of Cognition, Vol. 2, Psychological and Biological Models*. MIT Press, Cambridge (1986).
6. Rumelhart D., McClelland J. and PDP Research Group *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, Foundations*. MIT Press, Cambridge (1986).
7. Klimasauskas C. and Guiver J. NeuralWorks Networks I Revision 2.00, NeuralWare Inc., Sewickly, Pennsylvania.
8. Orlandea N. Node-analogous, sparsity-oriented methods for simulation of mechanical dynamic systems, Ph.D. Dissertation, The University of Michigan (1973).