Mathematical and Computational Applications, Vol. 12, No. 2, pp. 87-96, 2007.

THE USE OF ARTIFICIAL NEURAL NETWORKS IN SIMULATION OF MOBILE GROUND VEHICLES

Haldun Göktaş, Abdullah Çavuşoğlu, Baha Şen and İhsan Toktaş
Gazi University, Technical Education Faculty, Ankara,Turkey.
acavus@gazi.edu.tr

Abstract– In this study, we have developed a platform which incorporates Artificial Neural Networks (ANNs) in simulating body dynamics of mobile ground vehicles (e.g. cars). This is a part of our research project in which we plan to provide a platform for educating the driver candidates in virtual environments: where the drivers can be educated fully in "Artificial Cities". To start with, 6 different makes of cars with different engine properties has been simulated with the appropriate data provided by the manufacturers and rules of physics. A joystick steering wheel has been used to produce the necessary inputs for the ANN based physics engine. To train the network, Scaled Conjugate Gradient (SCG) and Levenberg-Marquardt (LM) learning algorithms and a logistic sigmoid transfer function have been used. The statistical error levels are negligible. The Absolute Fraction of Variance (R^2) values for both the training and test data are about 99.999% and the mean error value for both data group is lesser than 0.5%.
Keywords– Artificial Neural Networks, Driving Education, Car Simulation**.**

1. INTRODUCTION

Simulation is a powerful approach in educating people who are not acquainted with the real physical environment. Flight simulators are one of the most common simulation platforms used for this purpose. The idea of using similar systems for educating people to learn driving ground based vehicles is not new. However, due to its commercial nature it is hard to see any publications revealing the details of this matter. Using this approach, it is possible to use the simulation tools that help the pilots to be trained without endangering the planes. Similarly, driver candidates who have not used a car in a traffic as yet, maybe first trained by these simulation tools to get mastered in the traffic.

In this study, to be able to quicken the calculations involved to find the responses of vehicles against actions of the drivers, an Artificial Neural Network (ANN) based mechanism has been developed. For the sake of realism, a car simulation program must use a steering wheel and pedals hardware. Thus, candidate will use a similar hardware that exists in real automobiles. In addition, a physics engine that receives the data from steering wheel and generates the behavior of a car is needed.

The data set obtained from the physics engine is for several cars under different situations is recorded to be used to train and test the network. The physics engine is a function which performs the physical principles and car dynamics on the data flow from the steering wheel and pedals hardware and generates outputs like speed, engine rotation per minute and advanced distance.

The physics engine requires heavy calculations that in turn effects the amount of time required for the calculations to find the new position of the vehicle. Therefore, from the

programming point of view, the ability to perform real time image generation is limited. To overcome this, an ANN based mechanism is incorporated into the physics engine. The input data was supplied by the pedals hardware, which originally depends on the specifications of the simulated cars and the output is the speed, rotation of the engine in rpm, and the traveled distance based on x and y values.
The algorithmic mechanism provided in this study, can be used as a base to develop even faster (i.e. in the case of the platforms requiring such fast responses) games involving car simulations and platforms developed for traffic educations and finally military applications such as tank simulators. To calculate rigid body dynamics common formulas given in [1] has been used.

2. OPERATING CAR PHYSICS

The first force that affects the car when driving on a straight road is the Traction Force (*TF*). It is generated by the engine and transmitted to the wheels. While the torque delivered to wheels turns the wheels forward on the surface of the road, a frictional resistance occurs between the wheels and the road surface. This resistance operates opposite to movement direction of the car and called as Rolling Resistance (*RR*). Another important force on the car which appears is the friction between the car and the flow of air. Therefore the total force affecting the car is the sum of traction force that push the car forward and opposite forces, namely the *RR* and Air Resistance (*AR*). When the sum of *RR* and *AR* becomes equal to *TF*, then the net force on the car reaches to zero. In this case either the car moves with a constant velocity or it stops. If the *TF* is bigger than the other opposite forces, then the car accelerates. The formulae appearing in the proceeding part are mainly taken from [1, 2]. According to second law of Newton the acceleration of the car is:

$$a = \frac{F}{M} \tag{1}$$

where M is weight of the car, F is net force on the car. The velocity of the car (v) is found by integrating the acceleration of the car over the time:

$$v = v_0 + dt \times a \tag{2}$$

where the dt is a time interval that subsequent calls of the function which performs the car physics engine, in the same way the traveled distance can be found by integrating the velocity over the time as given by equation:

$$p = p_0 + dt \times v \tag{3}$$

If the physics engine function is called ten times in one second then dt is equal to 100 milliseconds. This time interval is multiplied by the calculated instant acceleration and added to the velocity which is calculated in previous time interval, and then the instant velocity is found. If the car was stationary at first and begins to move afterwards, then the previous velocity is zero. And if the instant acceleration is negative then the velocity will decrease. The net force on the car is calculated by summing the *TF*, *RR* and the *AR*. The *TF* is denoted as $F_{traction}$ while *AF* as F_{drag} and *RR* as F_{rr}.

$$F_{net} = F_{traction} - F_{drag} + F_{rr} \tag{4}$$

The *TF* of an engine is different for each different cars and they generates different amount of torque for each different speed of engine. So each engine has its own torque and power curve. This curve gives the information about amount of torque and power delivered by the engine at a given speed of engine. Engine speed is measured by the number of axle rotation per minute (rpm). The *RR* is expressed as in the Equation 5. Where the C_{rr} is a constant and given as 13 for the most commonly used radial wheels. As the equation illustrates, the *RR* is directly proportional to the velocity:

$$F_{rr} = - C_{rr} \times v \tag{5}$$

The *AR* is expressed in equation 6, where the C_d represents the frictional coefficient. This constant depends on the shape of the car and determined by the manufacturers after wind tunnel tests. *A* is the frontal area of the car and is approximately 2.2 m^2 for many of the cars. *Rho* gives the density of air and which is nearly 1.29 kg/m^3. The equation includes the square of the velocity. Therefore, increases in the speed causes the air resistance to increase more dramatically.

$$F_{drag} = 0.5 \times C_d \times A \times Rho \times v^2 \tag{6}$$

Until the engine torque is delivered to the wheels it passes the gear and the differential of the car. These mechanics multiplies the torque comes from engine by a factor of depending on the mechanics involved. Each car has different mechanical properties – listed in Table 1 for the cars involved in the simulations- that effect the relevant calculations. When passing these mechanical sections some of the produced power is lost. This is called as transmission efficiency. In Equation 7, T_{engine} denotes the torque of the engine at a given rpm. Where x_g is the gear ratio, x_d is differential ratio, *n* is the transmission efficiency and R_w denotes the wheel radius.

$$F_{traction} = T_{engine} \times x_g \times x_d \times n / R_w \tag{7}$$

Table 1. Car properties provided by the manufacturers

Car type	Transmission Rate					x_d	n	R_w (inch)	Weight (kg)	C_d	Torque (Nm)	Engine Speed
	G1	*G2*	*G3*	*G4*	*G5*							
Megane	3.73	2.0	1.32	0.97	0.79	3.87	0.8	14	1075	0.3	137	4000
Civic	3.46	1.87	1.24	0.91	0.76	4.11	0.8	14	1096	0.3	152	4300
Focus	3.58	1.93	1.32	0.95	0.76	3.82	0.8	14	1140	0.3	145	4000
Marea	3.91	2.16	1.48	1.12	0.89	3.81	0.8	14	1140	0.3	145	4000
Astra	3.73	2.14	1.41	1.12	0.89	3.74	0.8	15	1062	0.3	150	3600
Accent	3.61	2.05	1.37	0.97	0.78	4.05	0.75	13	946	0.3	136	3000

By changing the gear of the car, gear ratio is changed and the torque delivered from engine reaches to the wheels by times represented by this ratio. Increases in the gear cause decrease in gear ratio. This means that increases in the gear results a decrease at the transferred torque [1, 2].

3. THE DEVELOPED APPLICATION

The program gets the automobile specific data from a file which includes the data corresponding to a number of cars. The data consist of gear ratios, differential ratio, transmission efficiency, wheel size, car weight, frictional coefficient C_d and the maximum torque of the car. Using this data the program simulates different cars.

Initially we have included six different brands with different engine properties. Including more cars in this file shall give us the opportunity to simulate more cars.

The developed application window is shown in Figure 2. First set of fields are used to display the angle of steering wheel which is turned and the degree of pedals that are pressed. The electronical details of such peripherals which communicate with a PC may be found in [3]. The maximum angle steering wheel turns to the left or right side is 110 degrees and the maximum degree that the pedals pressed are 100 degrees. Pressing or releasing the buttons on the steering wheel is displayed in the second set of fields. The unprocessed data that comes from the steering wheel and pedals can be seen in the third field set. Forces on the car and the outputs of physics engine are displayed at fourth and fifth set of fields respectively.

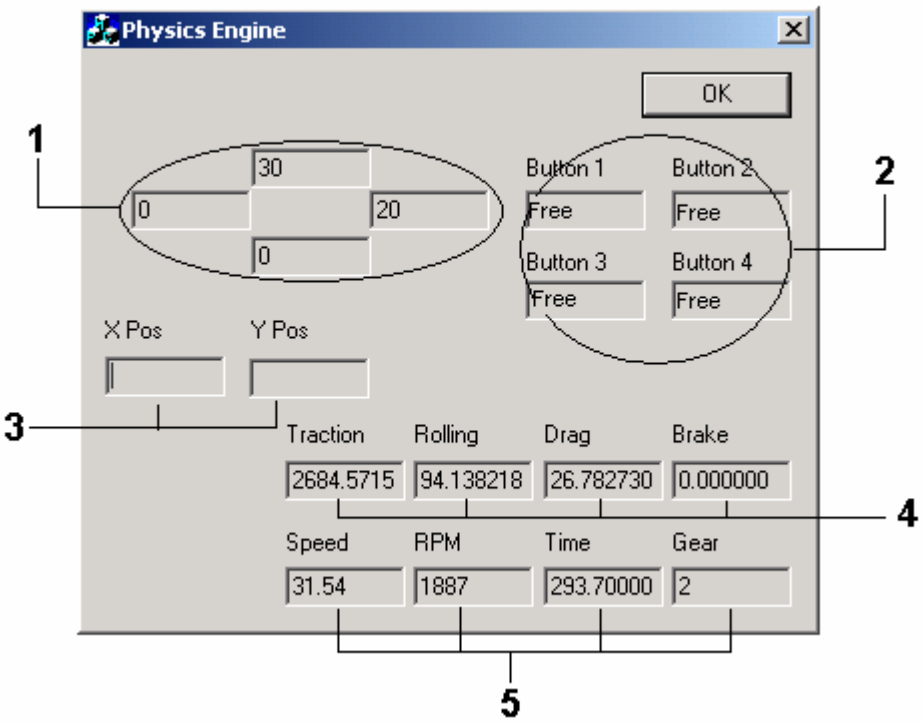

Figure 2. Snapshot of the Physics Engine program

This program runs as server side and graphics engine will run as client side. After connecting to the server, the client will get the information such as the instant speed of the car, rpm of the engine and *x, y* coordinates of the vehicle in the map. Since the graphics engine is not complete yet, to see the communication of the two programs, a temporary client application to resemble the graphics engine is developed. This application connects to the server which performs the physics engine and listens to the outputs coming from it. Temporary client application lists the incoming data in a field as can be seen in Figure 3. The client can send information to the server by pressing to the OK button. Physics engine sends instant speed of the car, instant value of rpm, distance that is advanced on the *y* and *x* axis to the client program with a "|" pipe character between each data. Each set of data that is send, once a time is displayed at the client side in one line and the following set is displayed in the next line. To send a data

in backward direction, (i.e. from client to server) a message is written to a smaller field in the client application and the OK button pressed. This message will be seen to the server in a message box.

3.1 The Performance of Physics Engine

To test the performance of physics engine generated outputs are compared by with real life values of simulated cars. Acceleration values from 0(zero) to 50 km/h, 100 km/h and 130 km/h are gathered from various sources such as car hand books, with the technical information like gear, differential ratios, car weight and so on. These technical values are written to the file which physics engine reads from to perform simulation. This information collection and performance test process is made for six different cars.

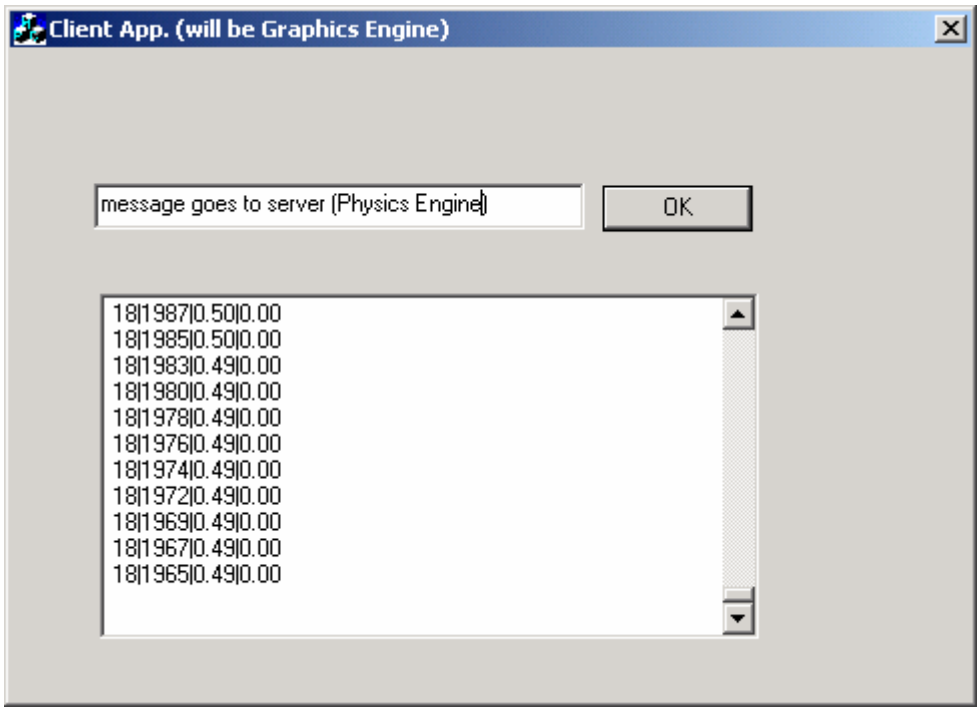

Figure 3. Snapshot of the client application (i.e. the graphics engine)

Real life values of acceleration time interval from 0 to 50 km/h, 0 to 100 km/h, 0 to 130 km/h and simulator results for each car are compared. Such a comparison for Honda Civic 1.6 LS is shown in Table 2. Maximum deviation ratio for the physics engine is about 6.3%. This amount is generally measured in milliseconds and the difference of a few milliseconds will not disturb the simulator user. The reason for the deviation is the lack of detailed technical specifications used for the cars. For example in some cases the values of C_d, the transmission efficiency and frontal area of the cars were hard to obtain, and in such cases approximate values are used, which in turn causes the deviation listed.

Table 2. Acceleration and test values for Honda Civic 1.6 LS

	Values in car magazine	Program generated value	Deviation ratio
0-50 km/h	3.4 s	3.2 s	5.8%
0-100 km/h	10.3 s	10.0 s	2.9%
0-130 km/h	17.4 s	18.5 s	6.3%

4. ARTIFICIAL NEURAL NETWORKS AND PREPARATION OF DATA

Our biological nervous system is simulated to solve complex functions in various applications. The ANNs are based on several nodes that are connected to each other. From the operational point-of-view, such a network operates as a black box: one side is used for inputs while the other side provides the required outputs. For this a training session is needed [4-11]. Several transfer functions [5-11] can be used to calculate outputs using normalized input values appropriately weighted. Fundamentally, there are two ANN learning models: supervised and unsupervised learning [6-11]. In this application, we have used supervised learning in which the network is presented with the inputs corresponding to the cars, along with the target outputs together with adjusted weighting.

One of the widely used learning-algorithms is the back-propagation algorithm. It has been used with a single hidden layer improved with numerical optimization techniques, namely: Scaled Conjugate Gradient (SCG), and Levenberg-Marquardt (LM) [7-11]. Inputs and outputs have been normalized in the range of (0-1). Neurons in the input layer have no transfer function, while in the other layers a logistic sigmoid (logsig) transfer function has been used and expressed as:

$$f(z) = \frac{1}{1+e^{-z}} \tag{8}$$

where the z is the weighted sum of the input. In order to train an artificial neural network, the experimental results have been used. The physical engine is experimented with each car to produce 1874 data outputs. This data is used for training the network while the remaining 40 are used for testing. At the input layer 10 variables are used namely: the acceleration time (T), car weight (M), wheel radius (R_w), friction (C_d), gear (G), gear ratio (x_g), differential ratio (x_d), air resistance (AR), rotation force (RR), and traction force (TF) while the output is the velocity of the car (v).
The computer program has been developed under the MATLAB. In the training session, we have used 3 different numbers of neurons (i.e. 6, 7, 8) in single hidden layer, for both SCG and LM. The test data is selected in equal numbers for each gear for each car for random acceleration times. The statistical methods of Root Mean Square error (*RMS)*, R^2, and mean % error values (*MAPE*) have been used for making comparisons. Figure 4 shows the one single hidden layer ANN architecture used in our application.

5. RESULTS

Table 3 summarizes statistical values corresponding to both the test data and training data with the predictions obtained from different learning algorithms –the LM and SCG- and varying hidden number of neurons. The LM algorithm with 7 neurons has produced the best results. The *MAPE* level in this algorithm is 0.459% in training and 0.504% in testing, R^2 values are 0.99% both for the training and testing and the *RMS* value is 0.286807 in the training and 0.3025 in the testing sessions. The results with the SCG (6 hidden numbers) produced maximum mean error percentages of 1.135% and 0.658% for training and testing respectively. Even these values are within the acceptable range considering the possible acceptable error ratio for the application.

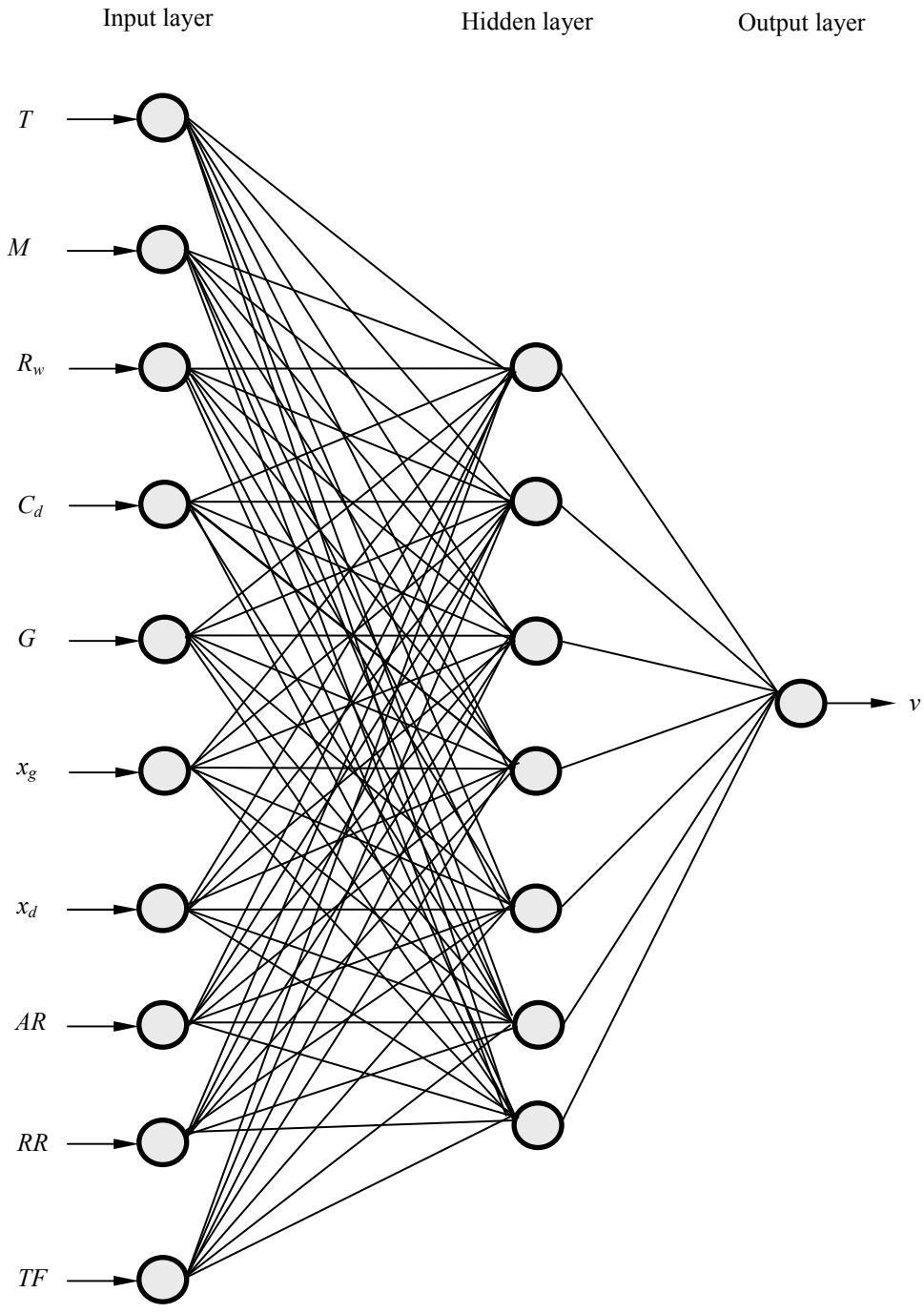

Figure 4. The ANN architecture used in the simulation

Table 3. Statistical values of predictions

	SCG6	*SCG7*	*SCG8*	*LM6*	*LM7*	*LM8*
RMS	0.434669	0.355286	0.413724	0.289263	0.286807	0.331906
R^2	0.999985	0.999990	0.999987	0.999994	0.999994	0.999992
MAPE	1.135873	0.854760	1.132149	0.482616	0.459997	0.771396
RMS-test	0.438950	0.362373	0.400230	0.294417	0.302500	0.348332
R^2*-test*	0.999976	0.999984	0.999980	0.999989	0.999989	0.999985
MAPE-Test	0.658459	0.575760	0.639169	0.480847	0.504329	0.466177

Figure 5 shows the results corresponding to test data while Figure 6 corresponds to the same for the both data. As the figures illustrate the actual and predicted velocity values are very close to each other.

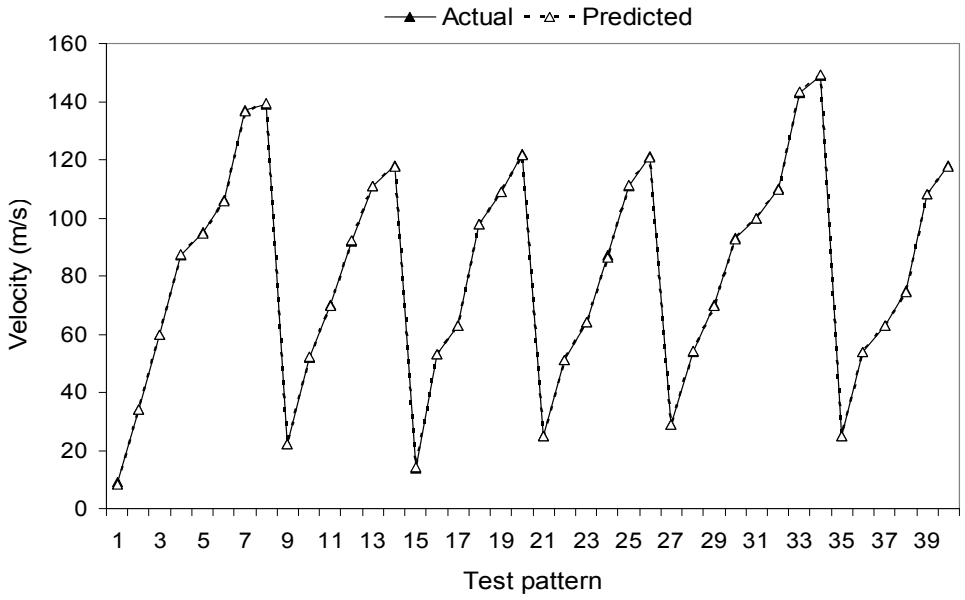

Figure 5. The actual and predicted values of velocity for the test data

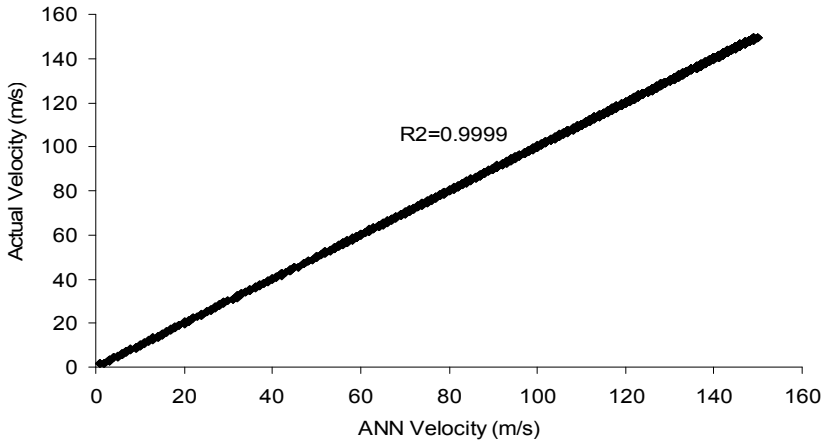

Figure 6. The actual and predicted values of velocity for the test and training data

The usage of the ANN system is often limited for standalone applications. For example we are considering several calculations which may be involved for finding the position of a mobile vehicle in a graphics based application. This naturally involves the calculations need to be done for each individual vehicle (i.e. apart from the 3D graphical calculations needed for the application). To give the system the ability to use the results produced by the ANN the following formulae have been extracted from the ANN system (using LM algorithm with 7 hidden neurons) makes the predictions easily calculatable and therefore removing the need to use the whole standalone ANN platform. As it may be noticed the number of arithmetical calculations involved in these formulae are much lesser then the ones needed for normal calculations (i.e. formulas) listed through Equation 1 to Equation 7. Table 4 provides the weights of neurons found in between the input and hidden layers. While the coefficients used in Equation 9 represent the weights between the hidden and output layers.

$$v = \frac{1}{1 + e^{-(1.8908\,F1 - 0.0302\,F2 - 0.0748\,F3 + 0.0119\,F4 - 5.9185\,F5 - 6.7632\,F6 + 6.7879\,F7 + 1.0795)}} \tag{9}$$

Table 4. The weights between input layer and hidden layer for LM 7

	$F_i = C_1 \times T + C_2 \times M + C_3 \times R_w + C_4 \times C_d + C_5 \times G + C_6 \times x_g + C_7 \times x_d + C_8 \times AR + C_9 \times RR + C_{10} \times TF + C_{11}$										
i	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}
1	-2.6294	-1.4131	1.9751	1.4476	2.7580	0.9896	-0.4088	-0.1203	7.1407	-1.3984	1.7046
2	-2.1380	-6.0312	0.7643	0.7064	1.0261	-4.3036	-2.3145	-0.5577	1.1799	0.6399	1.1506
3	2.5801	-2.3586	-1.7249	2.3032	-3.3014	-3.2146	-0.3875	1.4616	-1.2692	-4.4904	7.7797
4	-6.5763	-1.2094	-0.7444	0.6374	1.7297	-4.0540	3.2068	-5.6949	2.7562	-0.3215	5.7050
5	0.0115	0.0155	0.0241	-0.3266	0.0271	0.0593	0.0075	-0.0739	-2.2789	-0.0143	1.2095
6	-0.0581	-0.0156	0.1237	-4.1748	-0.0378	0.1499	0.0221	-4.2403	-0.8385	-0.2209	7.7376
7	2.4736	0.3210	-1.5335	0.0607	-0.3202	-1.0862	-0.0167	-2.4074	9.2423	1.1011	1.6001

The equation in Table 4 is dependent on the inputs of the network. When using this equation, the input values are normalized by using the Equation 10 was used:

$$Nor_{\%N} = 0.8 \times \left[\frac{\%N - \%N_{min}}{\%N_{max} - \%N_{min}} \right] + 0.1 \tag{10}$$

where $\%N_{min}$, are $\%N_{max}$ are minimum and maximum input values of all related data, $\%N$ is the value to be normalized. But, Equation 10 is not used for wheel radius and friction values because they are between 0 and 1. $\%N_{min}$, are $\%N_{max}$ values are given in Table 5.

Table 5. Maximum and minimum values for normalize used in Equation 10

	T	M	G	x_g	x_d	AR	RR	TF	v
N_{max}	40.9	1140	4	3.91	4.11	635.33	458.5	4862.84	149
N_{min}	0.1	946	1	0.97	3.74	0.06	4.32	1213.53	1

5. CONCLUSIONS

This study presents a mechanism which incorporates the use of artificial neural networks in simulations of several different cars with different properties successfully.

It also provides a mechanism to remove the need to use the ANN system as a standalone platform. The results also show that the predictions made by the ANN can safely replace the physics engine which has been simulated. This in turn reduces the number of calculations involved to determine the positions of the cars which in turn contributes by providing extra processor time for the calculations of real time graphical images. We believe that this system can be further developed to include more cars and other possible ground vehicles that are desired to be simulated. The presented study does not have a graphical interface, so it at present does not yet give the impression of going on the road in traffic like computer games or car simulators. However, this study is the outset of a whole platform. Further studies including the development of graphical interface is still going on. This will give users the opportunity of drive in a virtual city and apply all the traffic rules by improving the graphical part of the project and with the integration of these two physics and graphics parts. It will evaluate the user if driver candidate violates a traffic rule with a warning. Performance tests shows that physics engine runs satisfactory on the movement of straight line but it needs some enhancements about going on the crooked road.

Acknowledgements-This project has been supported by Turkish Republic Prime Ministry State Planning Organization (DPT). We wish to thank them for their valuable support without which this project could not be finished.

REFERENCES

1. S. Çetinkaya, Taşıt Mekaniği, Nobel Yayın Dağıtım, Ankara, Turkey, 1999.
2. B. Beckham, Physics of Racing Series, http://phors.locost7.info
3. K. James, PC Interfacing and Data Acquisition: Techniques for Measurement, Instrumentation and Control, Newnes, Elsevier Group, England, 2000.
4. S. A. Kalogirou, Applications of artificial neural-networks for energy systems, Applied Energy 67, 17-35, 2000.
5. A. Palau, E. Velo, L. Puigjaner, Use of neural networks and expert systems to control a gas/solid sorption chilling machine, International Journal of Refrigeration 22, 59-66, 1999.
6. D. D. Massie, Neural-network fundamentals for scientists and engineers, ECOS'01 4-6 July, Istanbul, Turkey, 2001.
7. Matlab Version 6.0 Online Manual.
8. A. Sözen, E. Arcaklıoğlu, M. Özalp, N. Çağlar, Forecasting based on neural network approach of solar potential in Turkey, Renewable Energy 30, 1075-1090, 2005.
9. İ. Toktaş, N. Aktürk, A New Approach Using Artificial Neural Networks for Conceptual Design of Cylindrical Helical Gears, MTET 2005 1st International Vocational And Technical Education Technologies Congress, Marmara University, İstanbul, Turkey, 754-762, September 5-7, 2005.
10. E. Arcaklıoğlu, Performance comparison of CFCs with their substitutes using artificial neural network, International Journal of Energy Research 28, 1113-1125, 2004.
11. A. Sözen, E. Arcaklıoğlu, M. Özkaymak, Turkey's Net Energy Consumption, Applied Energy 81, 209-221, 2005.