Proceedings of IMECE2006
2006 ASME International Mechanical Engineering Congress and Exposition
November 5-10, 2006, Chicago, Illinois, USA

IMECE2006-13072

APPLICATION OF NEURAL NETWORK TO FIND INITIAL STATE OF OPTIMIZATION PARAMETERS

A.Reza Babakhani
M.Sc Student
Department of Mechanical Engineering
Sharif University of Technology, Azadi Avenue
Tehran, Iran
Tel: (98)771-3533785
E-mail:rezababakhani59@yahoo.com

Hasan Sayyaadi
Assistant Professor
Department of Mechanical Engineering
Sharif University of Technology, Azadi Avenue
Tehran, Iran
Tel: (98)21-66165682
E-mail:Sayyaadi@sharif.edu

ABSTRACT

This paper derives an estimated function made by simple Neural Network to find initial state of optimization parameters. It changes a system of differential equations with boundary values to a system of equations with initial values. So a lot of time would be saved to solve it. As a result, the system with differential equations will reach the desired final state.

INTRODUCTION

Application of Neural Networks in optimization and solving differential equations with boundary value problems has made us use them to change the structure of some of the differential equations. In such conditions, the Neural Networks should be trained. In addition, solving the systems of differential equations with boundary value problems is a time wasting job. Isaac Elias et-al have proposed methods for boundary value problems with irregular boundaries by using Neural Networks[1].E.K.P.Chong proposed a class of Neural Networks to solve linear programming problems[2]. S. Ferrari proposed an algebraic approach for representing multidimensional nonlinear functions by feedforward Neural Networks [3]. In this paper, a single degree of freedom system is considered as the sample problem. The system should go to the desired final state and the objective function is that the energy used during the time should be minimized. By giving some arbitrary initial states of optimization parameters and solving the system, the final state of the system parameters will be found. Then by using a simple Neural Network and training it, an approximate relationship between the initial state of optimization parameters and the final state of the system parameters will be found. The data used for training the Net are the derived final state of the system parameters and the arbitrary initial state of optimization parameters as the Network input and the Network output respectively. Simulation results have shown that the trained Net can estimate the initial state that makes the system reach the approximately desired final state.

PROBLEM STATEMENT

In this paper, a single degree of freedom (1DOF) system is considered. The vehicle should go to the desired final state through an optimal path as shown in Fig(1).

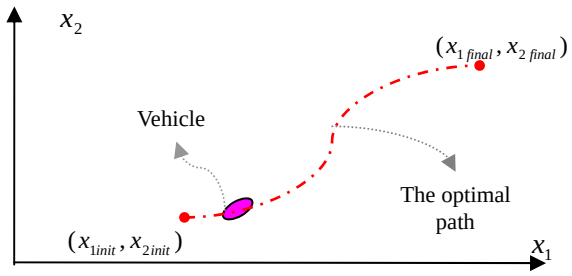

Fig.1: The sketch of the Vehicle in the state-spac

The vehicle has the following state-space equations:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{u}{m} \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{\left(x_2^2 + (x_1+1)\frac{u}{m}\right)x_3 + x_4 x_2 (x_1+1)}{x_3^2}\end{aligned} \quad (1)$$

Where x_1 is the vehicle's position, x_2 is the vehicle's velocity, x_3 and x_4 are auxiliary variables to derive a correct form of optimal input [4] . *m* represents the mass of the vehicle. u is the input of the system. The reason of using the auxiliary variables is that by getting the optimal input u and replacing it in the systems of equations, the optimal input will be a function of both the optimization and the system variables. By defining the equation:

$$x_3^2 = (x_1 + b)^2 \quad (2)$$

By differentiating two times of the equation to see the system's input, the other system equations will be found [5]:

$$\begin{aligned}\dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{\left(x_2^2 + (x_1+b)\frac{u}{m}\right)x_3 + x_4 x_2 (x_1+b)}{x_3^2}\end{aligned} \quad (3)$$

In this problem, it is assumed $b = 1$ wich is arbitrary. The objective function for path planning is energy to be minimized. So:

$$OF = \int_0^T u^2 dt \quad (4)$$

Where OF represents the objective function.

DYNAMIC PROGRAMMING METHOD

The method of optimization is dynamic programming. Dynamic programming is a kind of optimal control problem. The method is described as followed:
Consider the optimal control problem:

$$\min \phi(x(t_f)) + \int_{t_i}^{t_f} f_0(t, \vec{x}(t), \vec{u}(t))dt \quad (5)$$

Subjected to:

$$\begin{cases}\dot{\vec{x}}(t) = \vec{f}(t, \vec{x}(t), \vec{u}(t)) \\ \vec{x}(t_i) = \vec{x}_i \ , \ \vec{u}(t) \in U\end{cases} \quad (6)$$

Where t_i and t_f are fixed initial and final times and x_i is a fixed initial state. The end point $\vec{x}(t_f) = \vec{x}_f$ is free and can take any value in $\Re^n$. The control input is a piecewise continuous function, which satisfies the constraint $\vec{u}(t) \in U$, for $t \in [t_i, t_f]$.
The optimal function is defined as:

$$J^*(t_0, \vec{x}_0) = \min_{\vec{u}(.)} J(t_0, \vec{x}_0, \vec{u}(.)) \quad (7)$$

where the minimization is performed with respect to all admissible controls. This means in particular that the optimization problem (5) can be written:

$$J^*(t_i, \vec{x}_i) = \min_{\vec{u}(.)} J(t_i, \vec{x}_i, \vec{u}(.)) \quad (8)$$

The Hamiltonian is defined as:

$$H = \vec{y}^T \vec{f}(\vec{x}, \vec{u}) + OF \quad (9)$$

Where:
$\dot{\vec{x}} = \vec{f}(\vec{x}, \vec{u})$ represents the state equations of the system and $\vec{y} = [y_1, y_2, ..., y_n]$ is the vector of optimization parameters [6]. In addition, the following equations are considered with the systems of equations:

$$\dot{y}_i = -\frac{\partial H}{\partial x_i} \quad i = 1,2,...,n \quad (10)$$

The objective is that the Hamiltonian should be minimized to derive the optimal path. So the objective function would be minimized. In addition, the optimal inputs must be replaced by the equivalent variables. The optimal inputs are derived by using the following equations:

$$\frac{\partial H}{\partial u_j} = 0 \quad j = 1,2,...,k \quad (11)$$

Where k is the number of inputs. So the system's equations will be independent of the inputs [7].
In this example, the Hamiltonian is defined as:

$$H = y_1 x_2 + y_2 \frac{u}{m} + y_3 x_4 + y_4 \frac{(x_2^2 + (x_1+1)\frac{u}{m})x_3 - x_4 x_2 (x_1+1)}{x_3^2} + u^2 \quad (12)$$

By using the equation (11), the optimal input will be found and replaced in the system's equations. So the equations of the optimization problem are:

$$\dot{x}_r = f_r(\vec{x}, \vec{y}) \qquad \dot{y}_r = g_r(\vec{x}, \vec{y}) \quad r = 1,2,3,4 \quad (13)$$

$$g_r(\vec{x}, \vec{y}) = -\frac{\partial H}{\partial x_r} \quad (14)$$

In this problem, the initial state is:

$$x_{1init} = 0, x_{2init} = 0, x_{3init} = 1, x_{4init} = 0 \quad (15)$$

The relationship between the variables x_1 and x_2 and the variables x_3 and x_4 can be found by the equation (2). The objective is that the initial conditions of the optimization parameters $\vec{y}_{init} = [y_{1init}, y_{2init}, y_{3init}, y_{4init}]$ must be found to send the vehicle to the desired final state. A trained Neural Net is used to find the vector $\vec{y}_{init}$.

THE STRUCTURE OF THE NETWORK AND TRAINING

A simple Neural Net for training is used because of the simplicity of the example [8]. The MIMO Neural Net shown in Fig .2 is considered.

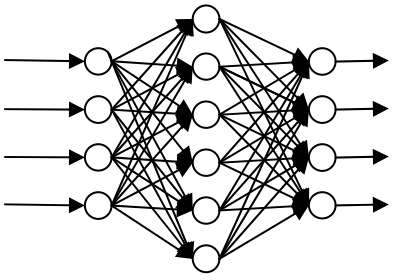

Fig.2: The structure of the Network

The relationship between the inputs and the outputs of the Network is shown in Fig .3.

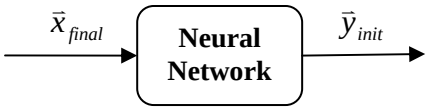

Fig.3: The relationship between the inputs and the outputs of the Network

The equations (13) are solved with arbitrary and different initial states of optimization parameters $\vec{y}_{init}$. The selected initial states have a domain that is predicted that the vehicle goes to the points around the desired final state by them. However, more different patterns can be selected to train the Net during a more extended domain It is solved 8 times and the time of the final state is 10 sec. The different patterns of $\vec{y}_{init}$ are shown in Table.1 and the derived final states are shown in Table.2. Each derived final state of the system parameters $\vec{x}_{final} = [x_{1final}, x_{2final}, x_{3final}, x_{4final}]$ is appropriate with the equivalent arbitrary initial state of optimization parameters $\vec{y}_{init} = [y_{1init}, y_{2init}, y_{3init}, y_{4init}]$. The vector with the values of (15) is considered as initial values for the system of equations (13). The cause of using the patterns of Table.1 is that the vehicle is wanted to go to the $\vec{x}_{dfinal}$ point. By using trial and error, the patterns that can cause the vehicle reach the points around the desired point $\vec{x}_{dfinal}$ will be found. These are the patterns that cause the Net make a relationship between the initial state of optimization parameters and the final state of the system parameters. It is impossible to estimate the initial state of optimization parameters $\vec{y}_{uinit}$ that cause the vehicle reach the desired final state $\vec{x}_{dfinal}$. If the number of patterns increased, the domain of training is more extended, so the vehicle can reach more desired final states. Fig.4 and Fig.5 show the domain of the vectors of the initial state of optimization parameters and the final state of the system parameters respectively. Each training input pattern $\vec{y}_{linit}$ makes the system reach the output training pattern $\vec{x}_{lfinal}$.

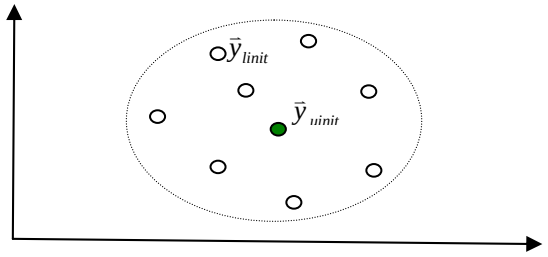

Fig.4:The vector space of the initial state of optimization parameter

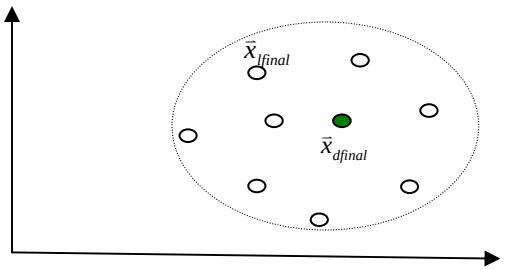

Fig.5:The vector space of the final state of the system parameters

The vehicle should go to the $\vec{x}_{dfinal}$ point and the pattern $\vec{y}_{uinit}$ that makes the system reach the point is unknown. The value of $\vec{y}_{uinit}$ is found by the trained Net.
The matrix of the Net weight of the hidden unit is:

$$W = \{w_{ij}\} \tag{17}$$

and the Net weight of the output unit is:

$$V = \{v_{jk}\} \tag{18}$$

that are initialized randomly.

The sigmoid functions of each neuron of the Net in Fig.2 are:

$$z_j = \frac{1}{1+\exp(-X_j)}, \qquad X_j = \sum_{i=1}^{4} w_{ij} x_i \tag{19}$$

for the hidden unit and:

$$y_k = \frac{1}{1+\exp(-Z_k)}, \qquad Z_k = \sum_{j=1}^{6} v_{jk} z_j \tag{20}$$

for the output unit respectively, where there are considered: i=1,2,3,4 for the input layer, j=1,2,3,4,5,6 for the hidden layer and k=1,2,3,4 for the output layer.

The data of Table.1 and Table.2 are considered as the inputs and the outputs for training the Net respectively. The method of training the Net is backpropagation. In fact, each pattern in Table.1 represents $\vec{y}_{linit}$ in Fig.4 and each pattern in Table.2 represents $\vec{x}_{lfinal}$ in Fig.5. l in the subscript represents the number of pattern.

Table.1: The arbitrary patterns to solve the system of equations

Number of Pattern	y_{1init}	y_{2init}	y_{3init}	y_{4init}
1	0.01	0.7	0.7	0.15
2	0.03	0.75	0.2	0.13
3	0.025	0.85	0.22	0.15
4	0.02	0.8	0.2	0.12
5	0.014	0.8	0.2	0.15
6	0.015	0.85	0.25	0.12
7	0.02	0.9	0.3	0.1
8	0.01	1	0.3	0.1
9	0.015	0.75	0.25	0.13
10	0.02	0.9	0.35	0.14
11	0.012	0.71	0.3	0.15
12	0.014	0.77	0.21	0.14
13	0.016	0.9	0.23	0.11
14	0.02	0.85	0.16	0.12
15	0.03	0.65	0.21	0.17
16	0.01	0.85	0.3	0.15

Table.2: The derived patterns of the final state of the system according to the data of table 1

Number of Pattern	x_{1final}	x_{2final}	x_{3final}	x_{4final}
1	-5.17	0.139	-4.17	0.139
2	-4.73	0.374	-3.73	0.374
3	-6.16	0.133	-5.16	0.133
4	-5.956	0.0174	-4.956	0.0174
5	-6.8	-0.22	-5.8	-0.22
6	-4.65	0.622	-3.65	0.622
7	-2.48	1.496	-1.48	1.496
8	-4.8	0.88	-3.8	0.88
9	-3	1.01	-2	1.01
10	-0.75	2.236	0.25	2.236
11	-0.3216	1.96	0.678	1.96
12	-5.576	0.134	-4.576	0.134
13	-6.32	0.1	-5.32	0.1
14	-8.84	-0.93	-7.84	-0.93
15	-3.14	0.82	-2.14	0.82
16	-2.98	1.31	-1.98	1.31

After training the Net, the trained data for the initial state of optimization parameters $\vec{y}_{init}$, are derived. The comparisons of the trained and the desired data for y_{1init}, y_{2init}, y_{3init}, and y_{4init} are shown in Fig.6, Fig.7, Fig.8, and Fig.9 respectively.

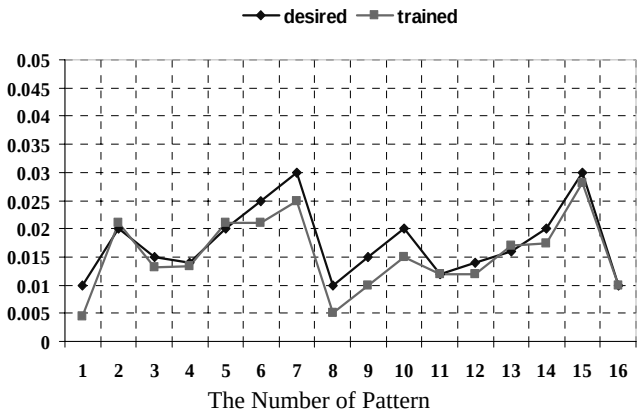

Fig.6: The trained values compared to the desired values for y_{1init}

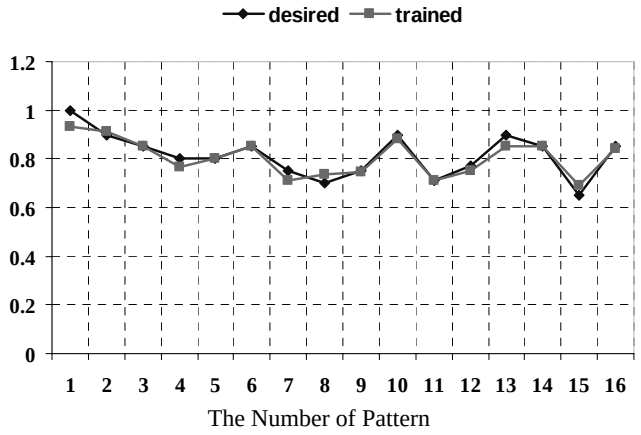

Fig.7: The trained values compared to the desired values for y_{2init}

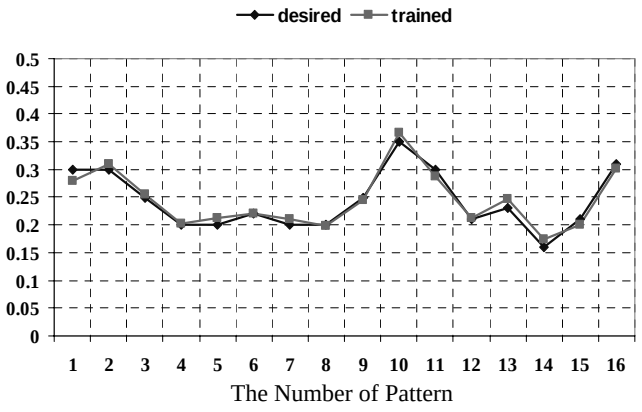

Fig.8: The trained values compared to the desired values for y_{3init}

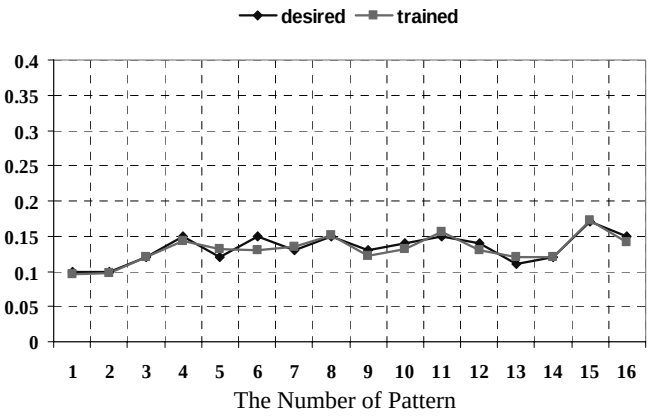

Fig.9: The trained values compared to the desired values for y_{4init}

After training, the vehicle is considered to go to the desired final state:

$$x_{1\,final} = -5, x_{2\,final} = 0.3, x_{3\,final} = -4, x_{4\,final} = 0.3 \qquad (21)$$

The data are given to the trained Net as inputs. The outputs of the Net are:

$$y_{1init} = 0.021, y_{2init} = 0.751, y_{3init} = 0.218, y_{4init} = 0.133 \qquad (22)$$

These data are added to the values(15) to be considered as the initial states to solve the equations (13) that is the proposed method. Another solution is derived by considering the values (13) and (21) as boundary conditions of the equation(13) to solve a boundary value problem that is the classical method.

As mentioned before, the trained Net in Fig.2 is used as an estimated function to find the initial state of optimization parameters.

The system of equations (13) is solved with $t_f = 10s$. Each time step equals to 0.33 s. The comparision of the solutions are shown in Fig.10 and Fig.11 for x_1 and x_2 respectively.

The method of solving the problems which can changes a boundary value problem to an initial value problem is called as the "Intelligent Converted Solution (ICS)". The thing in the solution is the amount of deviation of the system's final state from the desired that is affected by the quality of training the Net.

The time which is used to solve the initial value problem that is the ICS method is much shorter. The method which is used to solve the problem is nearly 8 times shorter than the classical method. On the other hand, the errors are replaced with the reduced computing time and cause the vehicle deviate from the desired final state.

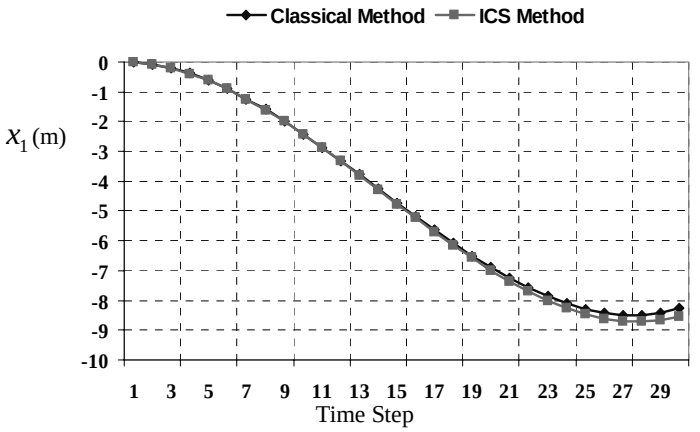

Fig.10: The comparison of the solutions for the vehicle's position versus time step

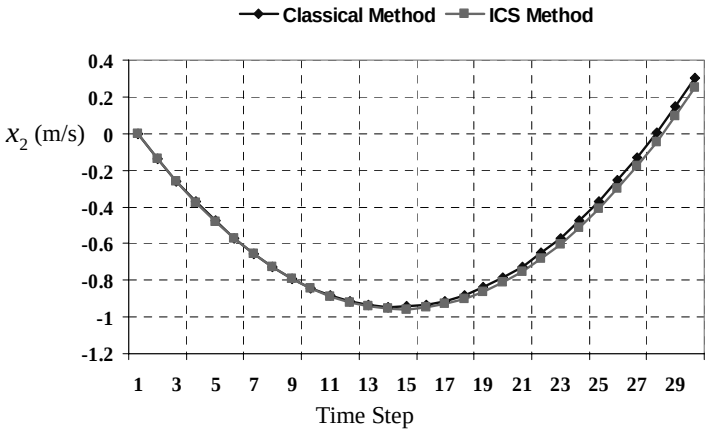

Fig.11: The comparison of the solutions for the vehicle's velocity versus time step

As it is shown in Fig.10 and Fig.11, it can be seen that the deviation from the desired final state of the system is 2.5% for x_1 and 8.7% for x_2. The better the training the net, the less the amount of the deviation is.

The sketch of the vehicle is shown in Fig.12 and Fig.13 in the state-space for the Classical method and the ICS method respectively.

In this paper, a part of the problem is solved with the classical method and the other part i.e. finding the initial state of optimization parameters is solved with an intelligent method, that's why the method is called as the "Intelligent Converted Solution (ICS)".

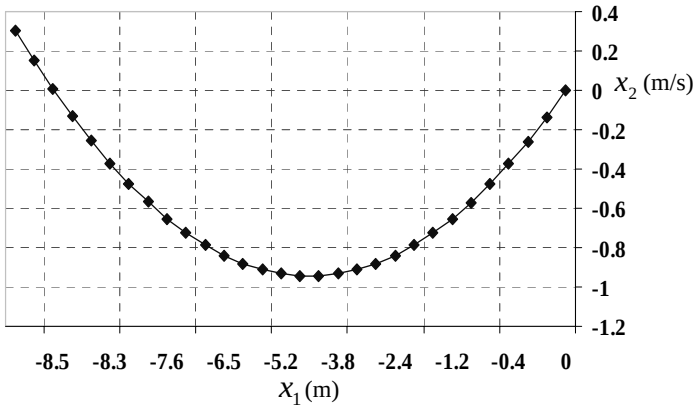

Fig.12: The sketch of the vehicle in the state-space, Classical Method

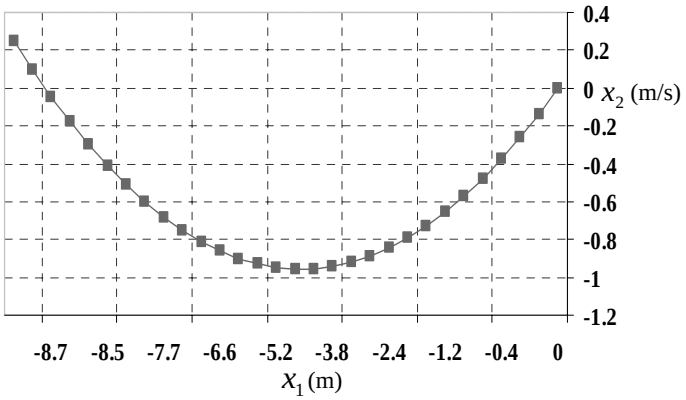

Fig.13: The sketch of the vehicle in the state-space, ICS Method

CONCLUSIONS

As a conclusion, it can be realized that by using Neural Networks, an approximate relationship between the initial state of optimization parameters and the final state of the system parameters can be found. So the system of equations can be solved by finding the initial states of optimization parameters. As a result, a lot of time will be saved. In the other words, the method is used for real time computation of differential equations caused by the optimal control problem. On the other hand, the errors are replaced to reduce the computing time of these boundary value problems. For the higher order systems, the domain of training and the number of patterns should be extended to consider more variations in the final state.

ACKNOWLEDGEMENTS

Acknowledgements to Roohollah Fatehi and Zahra Baba-khani for their assistance.

REFERENCES

[1]I.E. Lagaris, A.C. Likas, D.G. Papageorgeou, September 2000, "Neural Network Methods for Boundary Value problems with Irregular Boun-daries ", IEEE Transactions on Neural Networks, Vol.11, No.5, pp 1041-1049.

[2] E. K. P. Chong, S. Hui, S.H. Zak, November 1999, " An Analysis of a Class of Neural Networks for Solving Linear Programming Problems", IEEE Transactions on Automatic Control, Vol.44, No.11, pp 1995-2006.

[3] S. Ferrari, R.F. Stengel, Janurary 2005, " Smooth Function Approximation using Neural Networks", IEEE Transactions on Neural Networks, Vol.16, No.1, pp 24-38.

[4]A. R. Babakhani, January 2006, "Guidance and Control of AUVs while Moving Obstacles", MSc thesis, Sharif University of Technology, Tehran, Iran.

[5]Byron S.Gottfried, Joel Weisman, 1973, "Introduction to Optimization Theory", Department of Industrial Engineering, University of Pittsburg,.

[6] Unknown Author, 2003, "Formula Sheet for Optimal Control", Devision of Optimization and Systems Theory, Royal Institute of Technology, 10044 Stockholm, Sweden.

[7]Morton.M.Denn, 1969, "Optimization by Variational Methods", Department of Chemical Engineering, University of Delaware.

[8]Laurene Fausett, 1994, "Fundamentals of Neural Networks-", Florida Institute of Technology.