Piotr BOJARCZAK
Zbigniew GORYCA

# APPLICATION OF NEURAL NETWORKS TO ACCELERATION CONTROL OF ELECTRIC WHEELCHAIR

**ABTRACT** *In this paper the acceleration block of controlling software of electric wheelchair has been presented. This acceleration block was implemented with the use of MLP and neurofuzzy networks. For the sake of moderate throughput of microcontroller being used to implementation of this software, the network of smaller structure (neurofuzzy) has been chosen to realization.*

**Keywords:** *electric wheelchair, neural networks, acceleration control*

## 1. INTRODUCTION

Contemporary electric wheelchairs are fitted with brushless DC motors, its controlling system and joystick being used to determine the speed and direction of the movement. High power to mass ratio and noiseless operation of brushless

**Piotr BOJARCZAK, Ph.D.**
e-mail: bojarczp@amanda.radom.net
**Prof. Zbigniew GORYCA, Ph.D.**
e-mail: tgoryca@kki.net.pl
Radom University of Technology,
Malczewskiego 29, 26-600 Radom, POLAND
phone. +(48-48)361 77 10

PROCEEDINGS OF ELECTROTECHNICAL INSTITUTE, Issue 229, 2006

DC motors make designers go over to them. Presented here driving system of wheelchair consists of two brushless DC motors mounted separately in each wheel and corresponding controlling circuits. The controlling circuit in turn consists of circuit directly steering motors being called driver circuit and the microcontroller. The task of microcontroller is an assignment of the deflection of the joystick to the direction and the speed of the vehicle movement. Fig. 1 presents the workflow diagram of microcontroller. The microcontroller's program consists of three major parts.

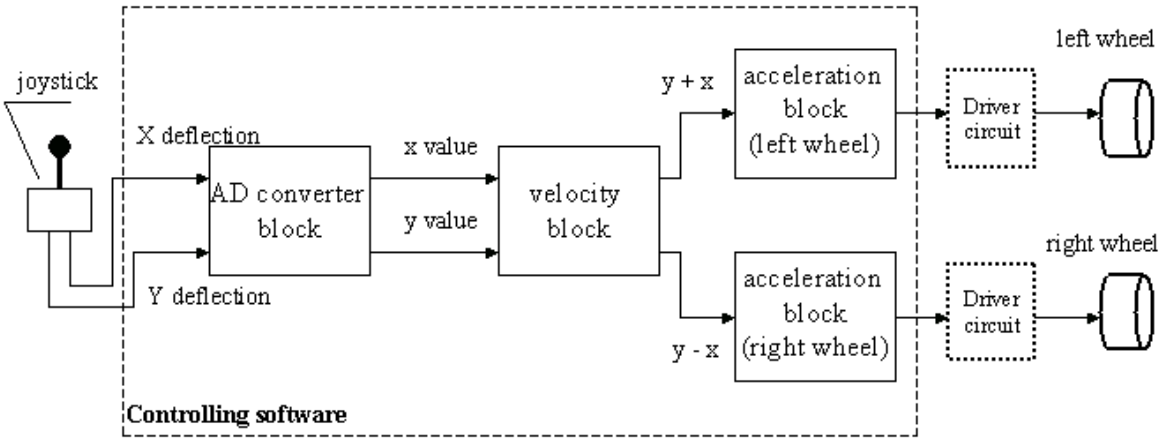


**Fig. 1. Structure of controlling system of electric wheelchair**

The first part is responsible for converting the deflection of the joystick (in $X$ and $Y$ axes) into its binary representation. It is achieved with the use of microcontroller's multi-channel AD converter. The converter's resolution allows obtaining two numbers – first for $X$-axis and the second for $Y$ in the range of –128 to 128.

In the second part being called velocity block, on the basis of previously determined binary representation of $X$ and $Y$ deflection, appropriate speed and the direction of the revolution for each wheel are calculated.

The acceleration block having already estimated speed and direction of each wheel determines acceleration needing to smoothly change the vehicle speed and direction. The last two parts deserve attention.

Let x and y be a binary representations of instantaneous joystick's deflection in $X$ and $Y$ axes respectively, then the right wheel velocity is equal to $y - x$ and the left wheel velocity is equal to $y + x$. Thanks to it, the joystick's deflection to the right corresponds to increasing the left wheel velocity and decreasing the

right wheel velocity – turning to the right. In the case of the joystick's deflection to the left, the left wheel velocity is decreasing and the right wheel velocity is increasing, what corresponds to turning to the left. In the third part of the program being called an acceleration block, on the basis of velocities obtained form velocity block and an actual instantaneous velocities the appropriate value of acceleration is calculated. The acceleration cannot be a constant. The value of acceleration should be dependent on the joystick's deflection. In the case of vehicle movement in the straight direction, the acceleration should be low at the beginning and increases successively in the further stage of the movement. It prevents from abrupt jerks during starting process. On the other hand the acceleration in turning phase should be much higher than in the straight direction phase, otherwise the vehicle will be unable to avoid the collision with obstacles.

Each acceleration block is assigned to every wheel. Therefore the value of acceleration should be the function of two variables – actual speed and the turning gauge being the difference between velocities of right and left wheels. On the basis of their experience and [1], authors decided to present the acceleration in the form shown in Fig. 2.

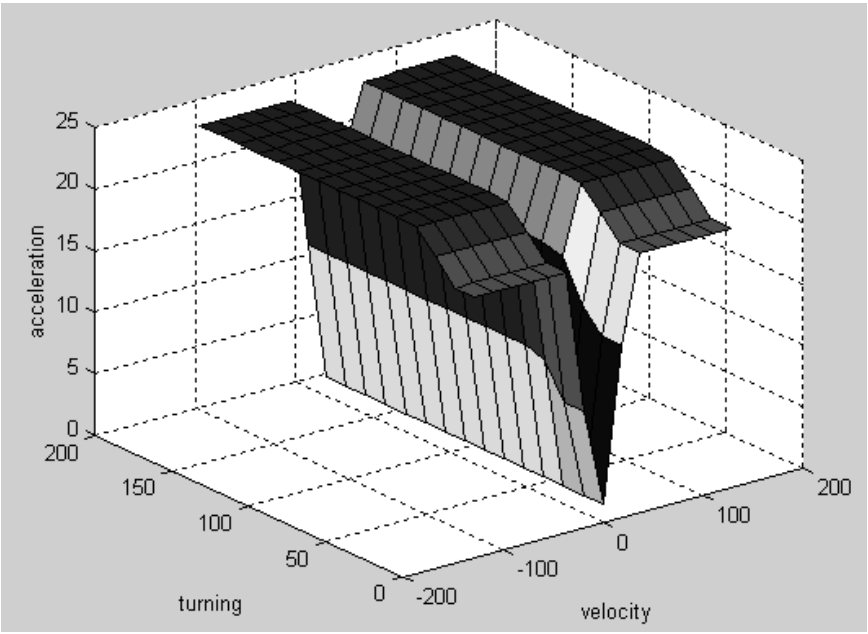


**Fig. 2. Dependence of acceleration on velocity and turning gauge**

# 2. IMPLEMENTATION OF ACCELERATION BLOCK IN THE FORM OF MLP NETWORK

In order to present the acceleration from Fig. 2 in the algebraically form the MLP (Multi- Layer Perceptron) network has been used. According to [2], on the basis of provided learning data, MLP is able to describe any relationship with any accuracy. Fig. 3 shows the structure of MLP network.

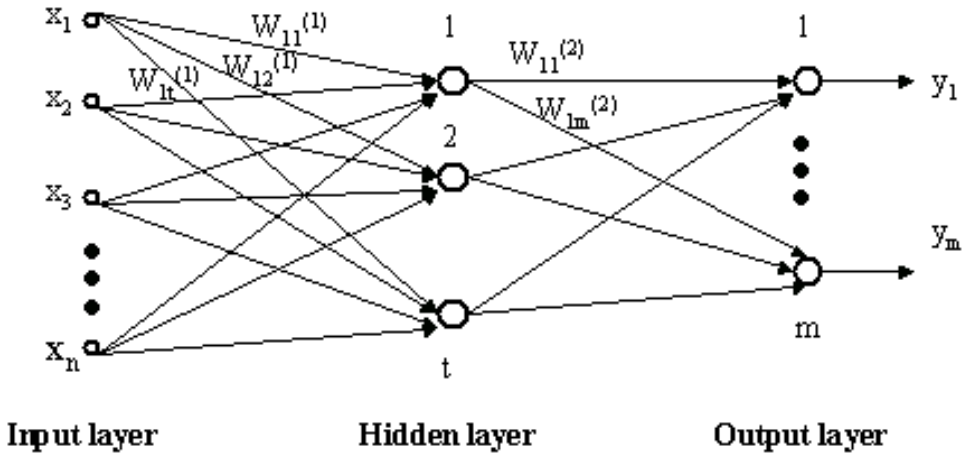


**Fig. 3. Structure of MLP network**

It consists of layers having neurons. The network of Fig. 3 has three layers being called input, hidden and output layers respectively. Only hidden and output layers have neurons. The task of input layer elements is the distribution of input vector components to the hidden layer's neurons. Both input layer elements and neurons of two last layers are connected through weights. Let $X = [x_1, x_2, ...., x_n]$ be an input vector given to the network's input and $D = [d_1, d_2, ...., d_m]$ be the destination vector given to the network's output, then the learning process consists in weights adaptation which leads to minimization of square error defined in following manner:

$$E = \sum_{k=1}^{p} \sum_{i=1}^{m} \left( y_i^{(k)} - d_i^{(k)} \right)^2 \tag{1}$$

where $p$ means the number of pairs of vectors $X$ and $D$.

There exist several methods being used to weights adaptation [ 2]. Authors resolved to choose backpropagation with momentum algorithm. After many experiments the network having 2 neurons in the input layer, 20 neurons in the hidden layer and 1 neuron in the output layer has been chosen. Neurons of the hidden layer have sigmoid activation function and neuron of the output layer has linear activation function. The structure of learning data consists of two vectors $X$ and D. Vector $X$ given to the input has two components, first corresponding to the velocity value and the second corresponding to turning gauge. Vector D given to the output has only one component corresponding to desired value of acceleration. The learning data having 250 pairs of $X$ and D vectors have been divided randomly into two groups, first consisting of 175 pairs and second consisting of 75 pairs. The first group was used to training the network and the second group to testing learned network. The network was constructed and learned with the use of NeuroSolution developing software. Fig. 4 shows the chart of acceleration produced by learned neural network. If we compare Fig. 1 and Fig. 4 we notice only slight difference between them. The main drawback of the solution being based on MLP network is a significant number of neurons needing to describe the acceleration relationship. Such large neural network structure makes its implementation in AVR microcontroller quite difficult.

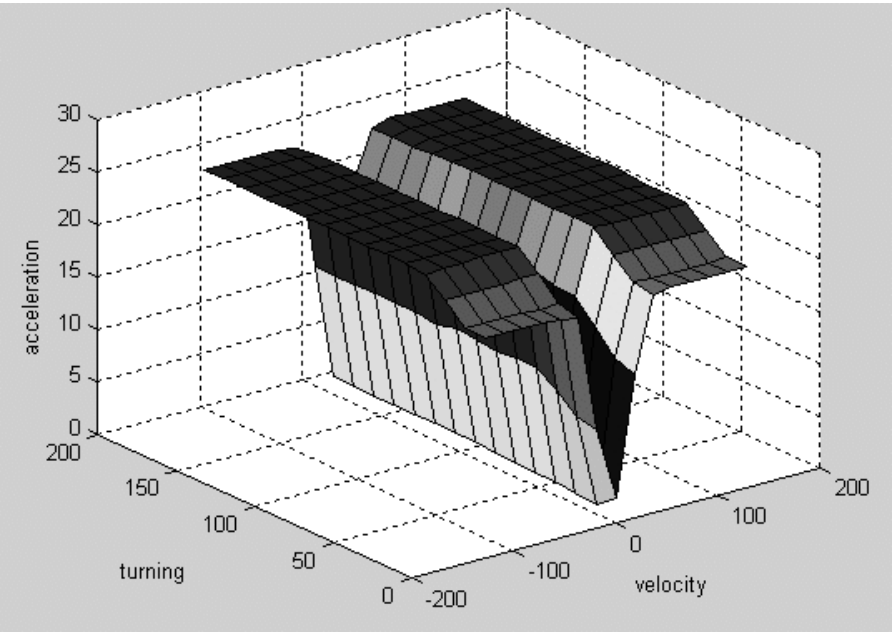


**Fig. 4. Acceleration relationship generated by learned MLP network**

# 3. IMPLEMENTATION OF ACCELERATION BLOCK IN THE FORM OF NEUROFUZZY NETWORK

As it can be noticed the structure of MLP network is large. In the second approach to the implementation of acceleration block, the neurofuzzy network has been used. The operation of this network is based on fuzzy set theory. In fuzzy sets, the degree of membership of element $x$ in appropriate set A is determined on the basis of membership function $\mu_A(x)$. The value of membership function is of range (0, 1). If value of membership function $\mu_A(x)$ is equal to 0, the element $x$ is not a member of the set A. If value of membership function $\mu_A(x)$ is equal to 1, the element $x$ is a member of the set A. When the membership function $\mu_A(x)$ has values between 0 and 1, element $x$ is partially contained in the set A. In fuzzy sets, $x$ is called the linguistic variable. In our case there are two linguistic variables – velocity $x_1$ and turning gauge $x_2$. First linguistic variable – velocity $x_1$ can take following linguistic values: "low" and "high" and second linguistic variable – turning gauge $x_2$ can take: "straight direction" and "turning". Therefore we have two fuzzy sets: "low" and "high" for the velocity variable and additional two fuzzy sets: "straight direction" and "turning" for the turning gauge variable. Each of these fuzzy sets has the own membership function. The relationship between acceleration and linguistic variables are described with the use of "if-then" rules. In the case of the neurofuzzy network being used in acceleration block this relationship is based on Takagi-Sugeno-Kanga (TSK) inference rules having the exemplary form:
**If** velocity $x_1$ is "low" and "turning gauge" $x_2$ is "straight direction" **then** acceleration $y = p_0 + p_1 x_1 + p_2 x_2$

where: $p_0, p_1, p_2,$ are coefficients whose values are adapted during the learning process. In the case of $M$ inference rules the acceleration formula takes the form:

$$y = \sum_{i=1}^{M} \frac{w_i}{\sum_{j=1}^{M} w_j} y_i = \sum_{i=1}^{M} w'_i \, y_i \tag{2}$$

where:

$$y_i = p_{i0} + p_{i1} x_1 + p_{i2} x_2 \tag{3}$$

is the acceleration relationship for the $i$-th inference rule. As we can see the overall acceleration relationship is equal to a weighted sum of all rules. The

meaning of $w_i$ are given in (5). According to [3, 4], the structure of the neurofuzzy network can be presented in the form of Fig.5. It consists of five layers.

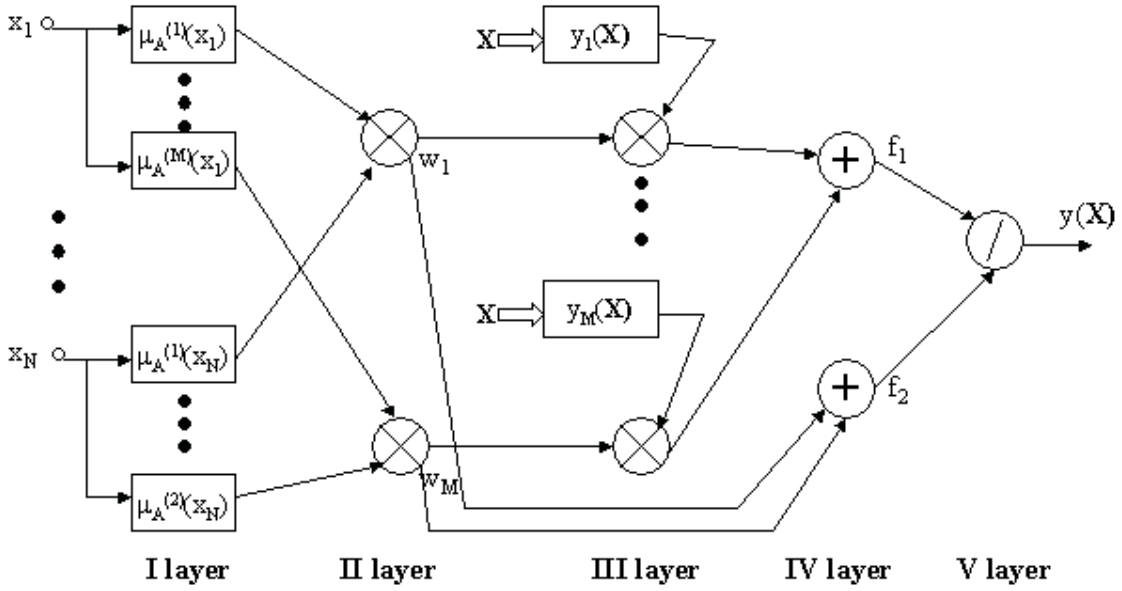


**Fig. 5. Structure of the pneurofuzzy network**

First layer contains the set of membership function of gaussian form:

$$\mu_A(x_i) = \frac{1}{1+\left(\frac{x_i - c_i}{\sigma_i}\right)^{2b_i}} \tag{4}$$

where: $c_i, b_i, \sigma_i$ are parameters whose values are adapted during the learning process.

Second layer calculates the weight $w_i$ according to (5) corresponding to the appropriate inference rule:

$$w_i = \prod_{j=1}^{N}\left[\frac{1}{1+\left(\frac{x_j - c_j}{\sigma_j}\right)^{2b_j}}\right] \tag{5}$$

where $N$ is equal to the number of linguistic variables, in our case $N = 2$.

For every inference rule third layer calculates the $y_i$ according to (3). Parameters $p_{i0}$, $p_{i1}$, $p_{i2}$ are adapted during the learning process.

Fourth layer contains two neurons. First calculates the weighted sum of $y_i$ and the second the sum of weights $\sum_{k=1}^{M} w_k$ .

The task of fifth layer (the last) is a simple division the value of $f1$ generated by the first neuron of fourth layer by the value of $f2$ generated by the second neuron of the fourth layer.

The structure and the number of learning data are the same as in the MLP network. After many experiments, 11 neurons of second layer have been chosen. NeuroSolution developing software has been used to construct and learn the network. The chart of acceleration being generated by neurofuzzy network is identical with this generated by MLP network (Fig. 4). Despite to its smaller structure, it is able to generate the acceleration relationship of the same accuracy as in MLP network.

# 4. CONCLUSIONS

The complexity of the acceleration block is a crucial topic when the controlling software is implemented on typical microcontroller having moderate throughput such as Atmel AVR. As it has been shown the complexity of structure of neurofuzzy network is much less than MLP network. The whole controlling software including acceleration block basing on neurofuzzy network has been written with the use of GNU C compiler for Atmel AVR microcontroller. The software was successfully tested in actual wheelchair prototype whose image is shown in Fig. 6.


**Fig. 6. The wheelchair prototype**

## LITERATURE

1. Fręchowicz A.: *Use of fuzzy logic in control speed circuit of a powered wheelchair,* Conference of Electric Traction, Zakopane, October, 2002 (in polish)
2. Haykin S.: *Neural networks, a comprehensive foundation,* Prentice Hall, New Jersey, 1999.
3. Nguyen H.T, Prasad N.R, Walker C.L, Walker E.A : *A first course in fuzzy and neural control,* CRC Press, Florida, 2003
4. Ross T. J.: *Fuzzy logic with engineering applications,* McGraw-Hill, USA, 1995

*Manuscript submitted 14.11.2006*
***Reviewed by Jerzy Zadrożny***

# ZASTOSOWANIE SIECI NEURONOWYCH DO STEROWANIA PRZYSPIESZENIA ELEKTRYCZNYCH WÓZKÓW DLA INWALIDÓW

P. BOJARCZAK, Z. GORYCA

**STRESZCZENIE** *Artykuł omawia oprogramowanie bloku sterowania przyspieszenia wózka inwalidzkiego. Blok ten wdrożono przy użyciu MLP i sieci neuronowej rozmytej. Dla moderacji przepustowości użytego mikrosterownika do wdrożenia tego oprogramowania wybrano do realizacji sieć o mniejszej konstrukcji (neuro-rozmytą).*