computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

APPLICATION OF OBJECT ORIENTED NEURAL NETWORK TO CONTROL MOTION OF THE LOAD OF A SEA CRANE

Pawel Falat [1], Lucyna Brzozowska [2], Krzysztof Brzozowski [3]

[1] University of Bielsko - Biala, ul. Willowa 2, 43-309 Bielsko-Biala, Poland, falat@ath.bielsko.pl
[2] University of Bielsko - Biala, ul. Willowa 2, 43-309 Bielsko-Biala, Poland, lbrzozowska@ath.bielsko.pl
[3] University of Bielsko - Biala, ul. Willowa 2, 43-309 Bielsko-Biala, Poland, kbrzozowski@ath.bielsko.pl

Abstract: *The paper presents object oriented approach to design of neural networks. The second part of the article presents an application of the object oriented neural network to control the load of the sea crane of an A-Frame type. The control algorithm has to stabilize load position and compensate the weaving. The model of the A-Frame dynamics were developed and used to achieve the optimal winch drive functions for various sea conditions. Those functions have been used to teach the network.*

Keywords: *Neural network ,object oriented programming, optimization, motion control, A-Frame crane, .NET.*

1. INTRODUCTION

The designing process of neural network can be quite complicated. The object oriented approach can be used for better understanding the net construction and behavior while creating the computer program.

2. ADVANTAGES OF OBJECT ORIENTED APPROACH

Use of object oriented system, enables us to describe physical and logical relationships between parts of the system. This means that the real world can be transferred "as it is" to the computer virtual environment. Because of that the whole system is easier to understand. Another advantage is the flexibility of such system. It is easy to add new functionality by adding another method to the object definition or creating derivative objects with different functionality than parent.

3. DISADVANTAGES (PROBLEMS) WITH OBJECT ORIENTED SYSTEM

There are also problems in designing the object oriented neural network system. The main problem is connected with the particular object design. Apart from its own functionality (such as display method or internal calculation functions) the object design must consider the interaction (connections) with other objects (different or of the same type). Second problem is how to save the whole network structure, especially how to remember the connections between the objects. This problem can be solved using .NET Framework objects (Collections) and mechanisms (Serialization).

4. OBJECT DESIGN

The object oriented approach makes it necessary to analyze and implement into virtual environment the entities from a real world.

The object structure is explained in the following figures. In Fig. 1. the basic neural objects structure is presented.

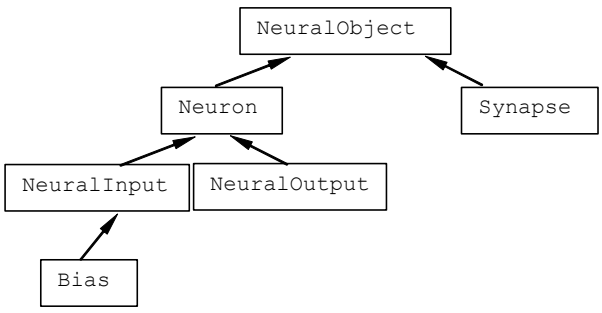

Fig. 1 – Hierarchy of classes

All objects are derived from a NeuralObject class which implements basic display functionality. Every class implements its own display methods and adds its own "neural" behavior.

The neuron (Fig. 2.) consist of two collections (objects of ArrayList type) which holds references to the Synapses. This kind of object is used to transfer the signal from other neurons (from other Layers) or

from Input objects.

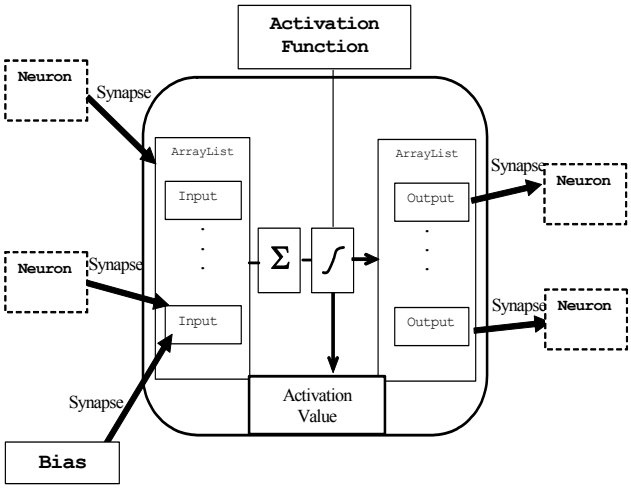

Fig. 2 – The Neuron object structure

The Neuron object calculate the sum of the signals which comes from Synapses and sends it to ActivationFunction object to calculate the ActivationValue. That value is used for outputs from the Neuron. As activation function can be used any object that implements the IActivationFunction interface. That gives the possibility for designer to develop different types of activation function.

The Synapse object (Fig. 3) is used as was mentioned to link the neurons with inputs or with other neurons. It consists of two references to Neuron Objects (Input and Output) and multiplication factor which is used during the signal transfer process from input to output. This design of Neuron and Synapse objects has some next advantages. It can be easily transformed to a network application when every neuron can exist in separate computer. The Synapse object would be then responsible for transfer of data between different machines.

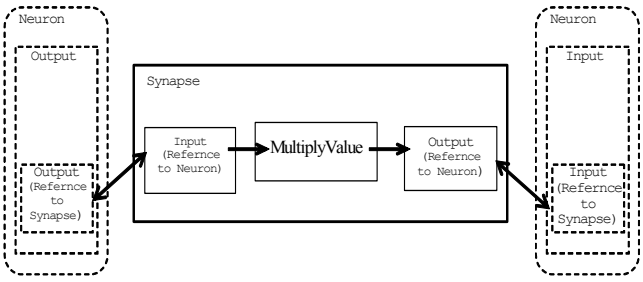

Fig. 3 – The Synapse object structure

In order logically connect the neurons in layers the NeuralLayer class has been created (Fig. 4). It collects the neurons and gives functionality to connect the neurons with other layers by creating the Synapses.

All those neural objects are combined to the net form in the NeuralNetwork object (Fig. 5). It gives functionality to calculate the network outputs for a given vector of input values.

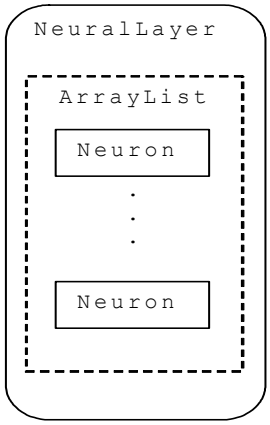

Fig. 4 – The NeuralLayer object structure

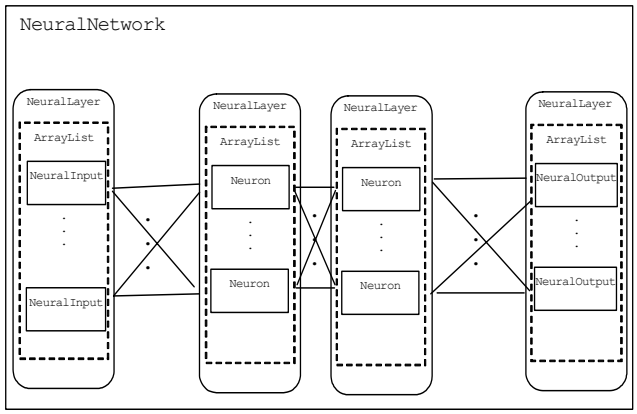

Fig. 5 – The NeuralNetwork object structure

Having created the network it must be tought. To do this the NeuralTeacher object can be developed. It should hold the reference to the NeuralNetwork object (Fig. 6) and by changing the multiplyValue inside the Synapse objects teach the network by using teaching algorithms. Until now only the backward propagation algorithm has been implemented [1] but work with the genetic type of teaching algorithms is in progress.

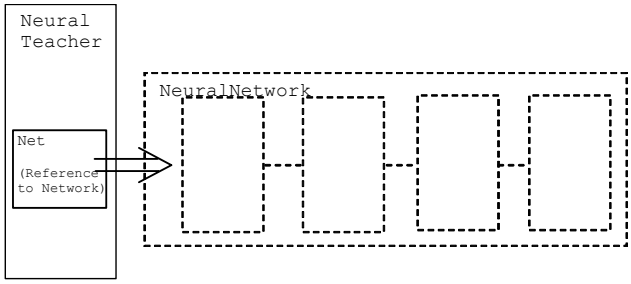

Fig. 6 –The NeuralTeacher object structure

Those are the most important objects which are used to create and use the neural network. They are summarized in the Table 1.

Table 1. Summary of object definition

Object Type	Function
Neuron	This is a primary object which represents the neuron behavior. It calculates the sum of inputs which comes from the collection of Synapse objects and send that value to the ActivationFunction object.
Synapse	The Synapse is used as a multiplication object for the signal which is transferred by it.
NeuralLayer	The NeuralLayer is a collection of the neurons which are connected by the Synapses.
NeuralInput	The NeuralInput object is in fact a modification of the neuron object definition. Hence it was crated as derivative type of the Neuron class. It is used to introduce the signal values to the network.
NeuralOutput	The NeuralOutput object was designed by similar way to NeuralInput as a derivative class of the Neuron.
NeuralNetwork	The NeuralNetwork is a collection of neural layers which holds the Neuron objects. It uses other Neural Objects (Neurons, Inputs, Outputs and Synapses) to calculate the network state.
NeuralTeacher	This object is linked with the network. It is used to teach the network. By implementing it the different method of teaching the network can be easily achieved.
ActivationFunction	The ActivationFunction object is used with the neuron. Every neuron has its own activation function object. It is possible to build the network in which behavior of a particular neuron can be different.
Bias	It is a special kind of neuron. It gives a constant input value equals 1 for the connected Neuron.

They are also some additional objects which are used to simplify the operations with network such as: TeachingSet object, different kind of activation function object e.g. UnipolarActivationFunction, BipolarActivationFunction etc. [1].

5. USING THE NETWORK

First the NeuralNetwork object need to be created. Having created the NeuralNetwork object the designer needs to add the inputs and outputs to it. Next the layers with required number of neuron on it have to be created. At the end the Synapses have to connect neurons between the layers (Fig. 7).

```
NeuralNetwork     NN= new NeuralNetwork();

                  NN.AddInput(new NeuralInput());
                  NN.AddInput(new NeuralInput());
                  NN.AddInput(new NeuralInput());

                  NN.AddOutput(new NeuralOutput());
                  NN.AddOutput(new NeuralOutput());
                  NN.AddOutput(new NeuralOutput());

                  NN.Width=600;
                  NN.Height=400;

                  NN.CreateLayer(5);
                  NN.CreateLayer(4);
                  NN.CreateLayer(5);


                  NN.ConnectLayers();
```

Fig. 7 – The NeuralNetwork generation

After that stage the network should be taught (Fig.8). In the following example the Teacher object can set the activation function for neurons inside the Network object.

```
bPTeacher= new    BackwardPropagationTeacher(NN);


bPTeacher.LoadTeachingSetsFromFile("TeachinSets.txt");

bPTeacher.SetNeuronsActivationFunction( new
           SigmoidBipolarActivationFunction(1));

bPTeacher.Teach();
```

Fig. 8 – The Neural Network teaching process

At the end of the teaching process the network can be used to solve a problem for which has been designed. However in order to transform it to other program we must save its state which will be explained in following chapter of the paper.

6. SAVING THE NETWORK

One of the main problems with the neural network created by using objects connected by references is how to save its state. This means values which are used to multiply the signal in the synapses and the whole structure of the network. This may be very difficult to programming with complicated networks (Fig. 9) .

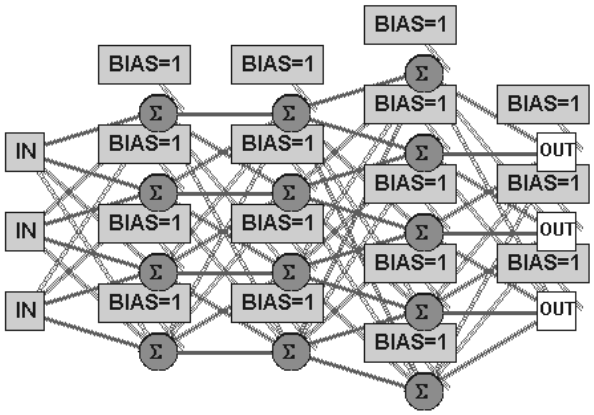

Fig. 9 – The Neural Network to save

In order to solve this problem the serialization mechanism has been used. The serialization is a process of saving the state of the object. It saves the values which are "in the object" and all objects connected with the object which is serialized as well. Hence the only one constraint which must be fulfilled is that all objects which are used to build the network must support the serialization mechanism. For .NET three types of Serialization are implemented [2]. XML Serializations, Binary and SOAP serialization. In the package presented only the last two has been implemented because XML serialization saves only data. It cannot save the connections between the objects, which is necessary in this case. The net can be saved in the binary file (binary serialization) or in text file which is used by SOAP serialization technology. The serialization mechanism have some constraints, for example it saves the name of assembly (*dll* or *exe* file) which carried out the serialization to the output file which can cause problems during deserialization process but it can be easily solved.

7. APPLICATION OF THE NEURAL SYSTEM

The system presented have been used to control the motion of a load hoisted by a offshore crane of an A-Frame type (Fig. 10). The task is to find the winch drive function which ensures stabilization of the load on proper depth. This can be done by using traditional optimisation methods which is presented in [3].

8. NEED OF NEURAL SOLUTION

The optimisation process is fast but not fast enough. It cannot be applied in real time. But it can be used to control effectiveness of other controlling algorithms. It can also be used to generate several teaching sets which can be used to teach an artificial controlling system as it can be the neural network. those sets can be obtained as a result of the optimisation process.

9. THE A-FRAME MODEL

The first step, to solve the problem, was to create a computer model of the A-Frame crane (Fig. 10). The important thing was to discover which construction properties are important for optimisation process. For that reason several models were implemented and tested [3],[4].

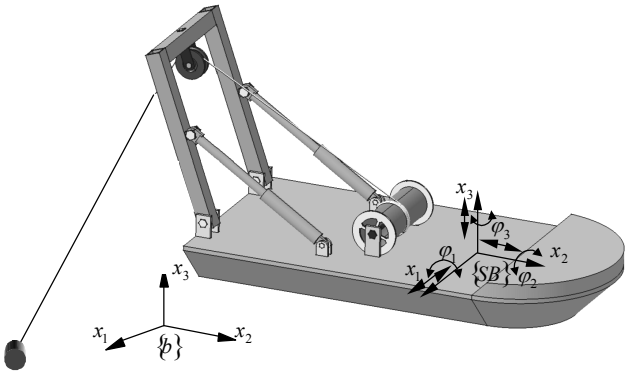

Fig. 10 – A-Frame Crane

The complete A-Frame dynamic model was created by using the Rigid Finite Element Method [3], which takes into consideration the flexibility of the structure and flexibility of the rope. The model created using ANSYS-ADAMS commercial packages, which have been used to control the own A-Frame model. For optimisation, which is used to find the winch drive function, stabilising the load on proper depth the separate model has been used. The model was implemented to reduce calculation time and does not take into consideration the flexibility of the construction. This is possible and was controlled in tests [3]. The flexibility of the rope must be taken into consideration and the optimisation model has included the flexible behaviour of the rope.

The parameters of ship hull movement and coordinates of the winch and cylinder position during derivation of the equations of motion of the probe were assumed to be known. Fig. 11 presents the model of the system which was optimised and values which have been used during calculations. The water damping ratio was not taken into account.

Ship motion was assumed to be known and is described by time functions

$$\begin{cases} x_C = x_C(t) \\ y_C = y_C(t) \\ z_C = z_C(t) \\ \varphi_x = \varphi_x(t) \\ \varphi_y = \varphi_y(t) \\ \varphi_z = \varphi_z(t) \end{cases}$$

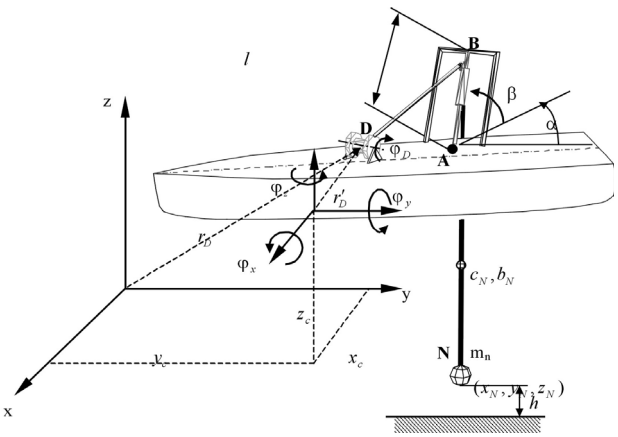

Fig.11- Scheme of the model

The equations of motion can be presented in a general form:

$$\begin{cases} m_N \ddot{x}_N + C_l \cdot \dfrac{\Delta l}{|BN|}\left(x_N - a_x - lb_x\right) = 0 \\ m_N \ddot{y}_N + C_l \cdot \dfrac{\Delta l}{|BN|}\left(y_N - a_y - lb_y\right) = 0 \\ m_N \ddot{z}_N + C_l \cdot \dfrac{\Delta l}{|BN|}\left(z_N - a_z - lb_z\right) + m_N g - F_W = 0, \end{cases}$$

where

l_0 - rope length without the load,

m_N - probe mass,

g - acceleration of gravity,

C_l - rope stiffness,

φ_D - angle of the winch drum

r_D - radius of the winch drum

$\Delta l = |DB| + |BN| - l_0 + \varphi_D \cdot r_D$

The vectors ***a*** and ***b*** from formulae have the form:

$$\boldsymbol{a} = \begin{bmatrix} x'_A - \varphi_z y'_A + \varphi_y z'_A + x_C \\ \varphi_z x'_A + y'_A - \varphi_x z'_A + y_C \\ -\varphi_y x'_A + \varphi_x y'_A + z'_A + z_C \\ 1 \end{bmatrix},$$

$$\boldsymbol{b} = \begin{bmatrix} -C_\beta S_\alpha - \varphi_z C_\beta S_\alpha + \varphi_y S_\alpha \\ -\varphi_z C_\beta S_\alpha + C_\beta S_\alpha - \varphi_x S_\alpha \\ \varphi_y C_\beta S_\alpha + \varphi_x C_\beta S_\alpha + S_\alpha \\ 1 \end{bmatrix},$$

$$C_\beta = \cos(\beta),$$

$$S_\alpha = \sin(\alpha).$$

The optimisation task is formulated: find winch angle $\varphi_D(t)$ so that, with these values known:

$$\varphi_x(t) \quad x_C(t)$$
$$\varphi_y(t) \quad y_C(t)$$
$$\varphi_z(t) \quad z_C(t)$$

the functional F defined below, is minimised:

$$F = \int_0^{t_K} [z_N - h]^2 \rightarrow \min$$

Having converted the optimisation task to a nonlinear one, the Nelder - Mead method, connected with the Powell method for searching for directional minima, was applied in order to solve it [6]. The complete model description can be found in [7].

10. NEURAL NETWORK

The network for this solution was created by using our own program written in C# for .NET platform by using the object oriented approach. The network was created as a classic net with 4 inputs, 3 hidden layers and 9 outputs (Fig. 12).

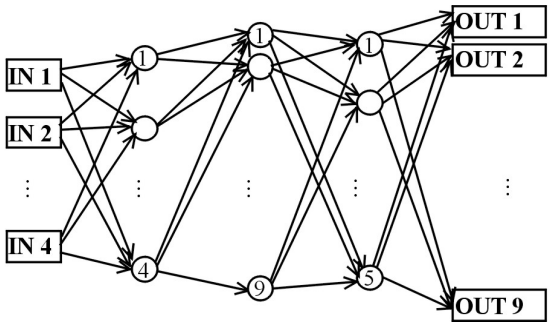

Fig.12- The network

The four ship motion parameters have been used as input values:

- the amplitude of the ship motion along z axis,
- the frequency of the motion along the z axis,
- the amplitude of rotations around the y axis,
- the frequency of rotations around the y axis.

Those parameters have the biggest impact on load behaviour. Every neuron in the network has its own activation function (sigmoid) and holds a reference to the bias object.

Output values are the parameters which describes the winch drive function (Fig. 13).

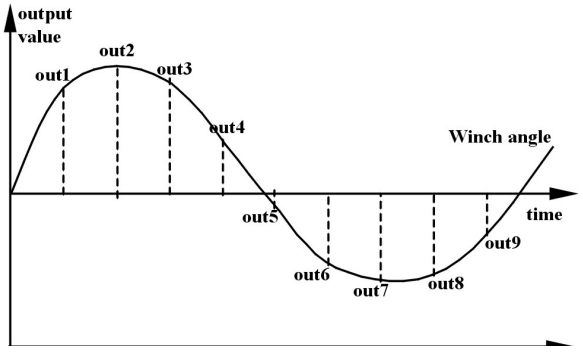

Fig. 13- The "neural" winch drive function

11. RESULTS

In the paper the first step of designing the neural control system is presented. In testing process some aspects have been discovered which need to be improved in the future.

For teaching process several (1400) teaching sets were generated by using the optimisation program.

The teachings sets cover wide spectrum of input parameters. The teaching sets have been applied in teaching process which use the backward propagation algorithm. After the test the net was saved and transferred to the program which simulates behaviour of the A-frame system. Then the testing simulation has been carried out.

Bellow the examples of calculation for the following input parameters are presented:

Table 2. The test parameters

Parameter	Case A	Case B	Case C
Amplitude Z [m]	2.25	2.5	2.6
Omega Z [rad/s]	0.52359	0.52359	0.50614
Amplitude φ_v [rad]	0.05934	0.12915	0.12915
Omega φ_v [rad/s]	0.366519	0.52359	0.52359

Two cases (Case A and B) show good results of neural control system. Case C is an example of improper results which doesn't ensure stabilisation of the load at the proper depth. Calculation results are presented on following figures:

Fig. 14 and Fig. 15 presents results for Case A,
Fig. 16 and Fig. 17 – Case B,
Fig. 18 and Fig. 19 – Case C.

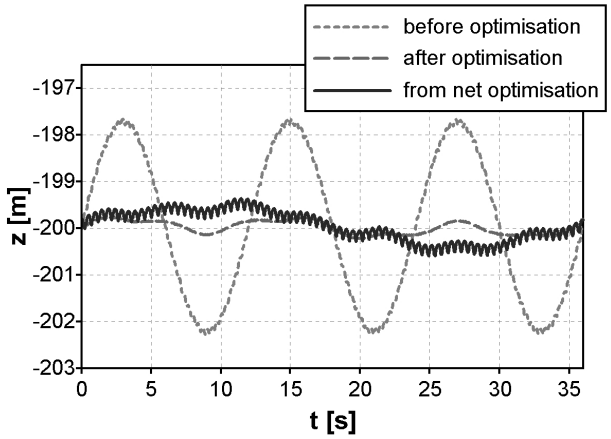

Fig.14- Case A: Load depth

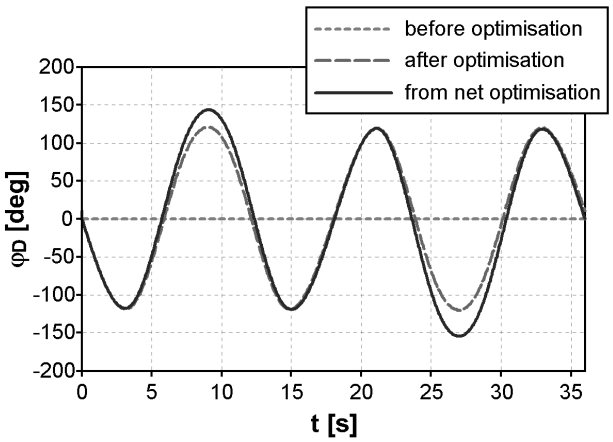

Fig.15- Case A: Winch angle

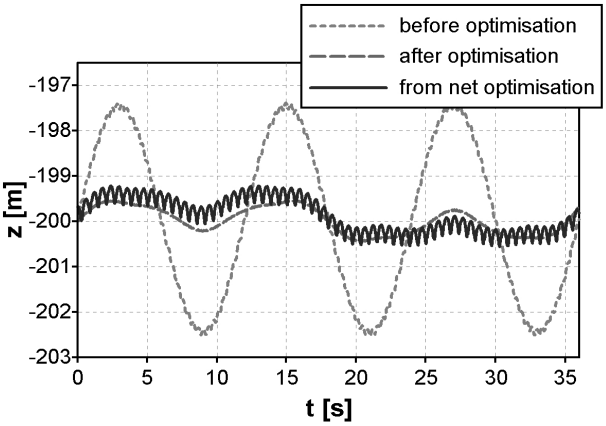

Fig.16- Case B: Load depth

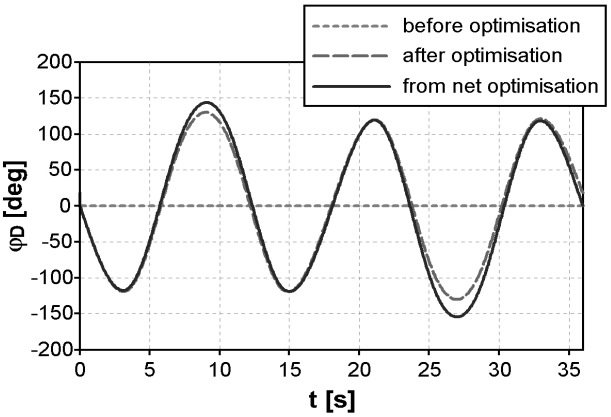

Fig.17- Case B: Winch angle

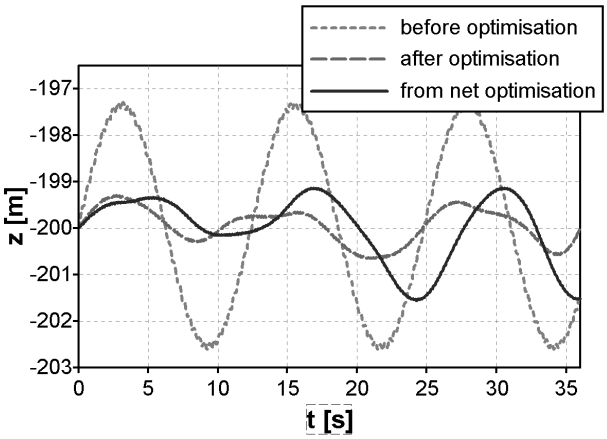

Fig.18- Case C: Load depth

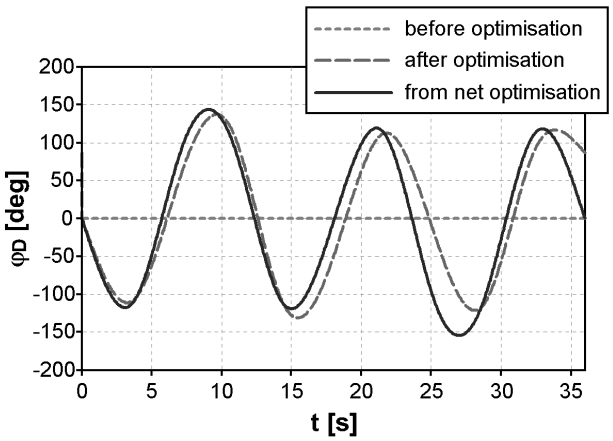

Fig.19- Case C: Winch angle

The best results were achieved for parameters which were placed at the end of teaching set file. This means that network "remembers" best what it learned at the end. That leads to a conclusion that the net should not be taught sequentially (set by set) but in random order or the teaching sets should covers wide spectrum of input parameters which neural network cannot remember.

12. CONCLUSIONS

The system presented is a first version of a package which is developed in department of Mechanics and Computer Methods. It is planed that package will implement function of visual design of neural network, design of different types of neural networks (e.g. radial), different types of teaching methods (in e.g. by using genetic algorithms).

In most considered cases the network gives acceptable solution. The winch drive function generated by network is effective and gives good load stabilisation. Unfortunately there are some cases for which huge error causes in the stabilisation process. That problem can be solved by using teaching sets which cover smaller spectrum of cases. This means that several nets which are specialized for specific range of parameters should be created and those networks should be used according to the input range. After that proper neural network would be loaded to neural controller as a program.

The teaching process also should be improved by implementing other teaching algorithms and other types of neural networks.

Another factor which can improve solution is change the type of output function for the optimization process and next for neural network. Because all input functions are periodical, the output function can be periodical as well an it can be modeled in a form:

$$\varphi_D = A\sin\left(\omega t + \varphi_0\right)$$

In that case the network will be searching parameters A, ω, φ_0. That might simplify the task and perhaps gives better results. This is are problems which will be investigated in the future.

13. REFERENCES

[1]. http://nrn.prv.pl/

[2]. http://msdn.microsoft.com/

[3]. P. Falat. *PhD Thesis: Dynamic analysis of an A-Frame crane,* University of Bielsko–Biala 2004

[4]. I. Adamiec-Wójcik, P. Fałat, T. Gancarczyk. *Computer Analysis of static loads of an A-Frame*, Zeszyty Naukowe Akademii Techniczno – Humanistycznej w Bielsku – Białej. Zeszyt nr 6, 2003, pp. 7-25

[5]. J. Kruszewski ,S. Sawiak ,E. Wittbrodt. *Metoda sztywnych elementów skończonych w dynamice konstrukcji,* WNT – Warszawa 1999

[6]. S.C. Chapra and R.P. Canale. *Numerical methods for engineers*, McGraw-Hill Higher Education. New York, 2002

[7]. P. Fałat, S. Wojciech. *Application of non-linear optimisation methods to stabilise motion of a sea probe*. Zeszyty Naukowe Akademii Techniczno – Humanistycznej w Bielsku – Białej. Zeszyt nr 6, 2003, pp. 29-40.

Paweł Fałat

Born in 1973,

Education:

B.Sc. (1997) and M.Sc. (1999) in Computer Methods for Mechanical Engineering at the Technical University of Lodz Branch in Bielsko–Biala. PhD (2004) at the Mechanical Engineering and Computer Science Faculty of the University of Bielsko – Biala.

MCAD (2004) Microsoft Certified Application Developer in .NET technology.

Position: Lecturer on subjects: C++, Java, C#, Programming for Internet, Computer Methods.

Areas of Interest: Programming languages, Web application, Web Services, Computer methods.

Lucyna Brzozozwska

Born in 1973,

Education:

B.Sc. (1997) and M.Sc. (1998) in monitoring and computer methods at the Technical University of Lodz Branch in Bielsko – Biala. PhD (2002) at the Faculty of Textile Engineering and Environmental Protection of the University of Bielsko – Biala.

Position: Lecturer on computer systems, monitoring and process control, database.

Areas of Interest: Computer systems, monitoring and process control.

Krzysztof Brzozowski

Born in 1972,

Education:

B.Sc. (1997) and M.Sc. (1998) in monitoring and computer methods at the Technical University of Lodz Branch in Bielsko – Biala. PhD (2002) at the Mechanical Engineering and Computer Science Faculty of the University of Bielsko – Biala.

Position: Lecturer on programming languages (Pascal, C++, Java), numerical methods.

Areas of Interest: Numerical methods, Computational Fluid Dynamics.