7

Underwater Robot Intelligent Control Based on Multilayer Neural Network

D. A. Oskin[1], A. A. Dyda[1], S. Longhi[2] and A. Monteriù[2]

[1]Department of Information Control Systems, Far Eastern Federal University Vladivostok, Russia
[2]Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Ancona, Italy
Corresponding author: D. A. Oskin <daoskin@mail.ru>

Abstract

The chapter is devoted to the design of an intelligent neural network-based control system for underwater robots. A new algorithm for intelligent controller learning is derived using the speed gradient method. The proposed systems provide robot dynamics close to the reference ones. Simulation results of neural network control systems for underwater robot dynamics with parameter and partial structural uncertainty have confirmed the perspectives and effectiveness of the developed approach.

Keywords: Underwater robot, control, uncertain dynamics, multilayer neural network speed gradient method.

7.1 Introduction

Underwater Robots (URs) promise great perspectives and have a broad scope of applications in the area of ocean exploration and exploitation. To provide exact movement along a prescribed space trajectory, URs need a high-quality control system. It is well known that URs can be considered as multi-dimensional nonlinear and uncertain controllable objects. Hence, the design procedure of URs control laws is a difficult and complex problem [3, 8].

Advances in Intelligent Robotics and Collaborative Automation, 147–166.

Modern control theory has derived a lot of methods and approaches to solve appropriate synthesis problems such as nonlinear feedback linearization, adaptive control, robust control, variable structure systems, etc [1, 4]. However, most of these methods for control systems synthesis essentially use information about the structure of the URs mathematical model. The nature of the interaction of a robot with the water environment is so complicated that it is hard to get the exact equations of URs motion. A possible way to overcome control laws synthesis problems can be found in the class of artificial intelligence systems, in particular, based on multi-layer Neural Networks (NNs) [1, 2, 5].

Recently, a lot of publications were devoted to the problems of NNs identification and control, starting from the basic paper [5]. Many papers are associated, in particular, with applications of NNs to the problems of URs control [1, 2, 7].

Conventional applications of multi-layer NNs are based on preliminary network learning. As a rule, this process is the minimization of the criterion which expresses overall deviations of NN outputs from the desirable values, with given NN inputs. The network learning results in NN weight coefficients adjustment. Such an approach supposes the knowledge of teaching input-output pairs [5, 7].

The feature of NNs application as a controller consists in the fact that a desirable control signal is unknown in advance. The desired trajectory (program signal) can be defined only for the whole control system [1, 2].

Thus, the multi-layer NNs application in control tasks demands a development of approaches that take into account the dynamical nature of controllable objects.

In this chapter, an intelligent NNs-based control system for URs is designed. A new learning algorithm for an intelligent NN controller, which uses the speed gradient method [4], is proposed. Numerical experiments with control systems containing the proposed NN controller were carried out in different scenarios: varying parameters and different expressions for viscous torques and forces. Modeling results are given and discussed.

Note that the choice of a NN regulator is connected with the principal orientation of the neural network approach to a priori uncertainty, which characterizes any UR. In fact, matrices of inertia of the UR's rigid body are not exactly known, as well as the added water mass. Forces and torques of viscous friction are unknown and uncertain functional structure parameters. Hence, an UR can be considered as a controllable object with partial parameter and structure uncertainties.

7.2 Underwater Robot Model

The UR mathematical model traditionally consists of differential equations describing its kinematics

$$\dot{q}_1 = J(q_1)q_2 \tag{7.1}$$

and its dynamics

$$D(q_1)\dot{q}_2 + B(q_1, q_2)q_2 + G(q_1, q_2) = U, \tag{7.2}$$

where $J\,(q_1)$ is the kinematical matrix; q_1, q_2 are the vectors of generalized coordinates and body-fixed frame velocities of the UR; U is the control forces and torques vector; D is the inertia matrix taking into account added masses of water; B is the Coriolis – centripetal term matrix; G is the vector of generalized gravity, buoyancy and nonlinear damping forces/torques [3].

The lack a priori knowledge of the mathematical structure and the parameters of the UR model matrices and the UR model vectors can be compensated by an intensive experimental research. As a rule, this way is too expensive and takes a long time. One alternative approach is connected with the usage of the intelligent NN control.

7.3 Intelligent NN Controller and Learning Algorithm Derivation

Our objective is to synthesize an underwater robot NN controller in order to provide the UR movement along a prescribed trajectory $q_{d1}\,(t)$, $q_{d2}(t)$.

Firstly, we consider the control task with respect to the velocities $q_{d2}\,(t)$. Let us define the error as:

$$e_2 = q_{d2} - q_2 \tag{7.3}$$

and let's introduce the function Q as a measure of the difference between desired and real trajectories:

$$Q = \frac{1}{2}e_2^T D e_2, \tag{7.4}$$

where the matrix of inertia is $D > 0$.

Furthermore, we use the speed gradient method developed by A. Fradkov [4]. According to this method, let compute the time derivative of Q:

$$\dot{Q} = e_2^T D\dot{e}_2 + \frac{1}{2}e_2^T \dot{D} e_2. \tag{7.5}$$

From

$$q_2 = q_{d2} - e_2 \tag{7.6}$$

and one has

$$D(q_1)\dot{q}_2 = D(q_1)\dot{q}_{d2} - D(q_1)\dot{e}_2. \tag{7.7}$$

Using the first term of the dynamics Equation (7.2), one can get the following:

$$\begin{aligned} D(q_1)\dot{e}_2 = D(q_1)\dot{q}_{d2} + B(q_1, q_2)q_{d2} - \\ - B(q_1, q_2)e_2 + G(q_1, q_2) - U, \end{aligned} \tag{7.8}$$

and thus the time derivative of function Q can be written in the following form:

$$\begin{aligned} \dot{Q} = e_2^T(D(q_1)\dot{q}_{d2} + B(q_1, q_2)q_{d2} - \\ - B(q_1, q_2)e_2 + G(q_1, q_2) - U) + \frac{1}{2}e_2^T\dot{D}e_2. \end{aligned} \tag{7.9}$$

After mathematical manipulation, one gets

$$\begin{aligned} \dot{Q} &= e_2^T(D(q_1)\dot{q}_{d2} + B(q_1, q_2)q_{d2} + G(q_1, q_2) - U) - \\ &- e_2^T B(q_1, q_2)e_2 + \frac{1}{2}e_2^T\dot{D}(q_1)e_2 = \\ &= e_2^T(D(q_1)\dot{q}_{d2} + B(q_1, q_2)q_{d2} + G(q_1, q_2) - U) + \\ &+ e_2^T(\frac{1}{2}\dot{D}(q_1) - B(q_1, q_2)e_2). \end{aligned}$$

As known, there is a skew-symmetric matrix in the last term, hence, this term is equal to zero, and we obtain the following simplified expression:

$$\dot{Q} = e_2^T(D(q_1)\dot{q}_{d2} + B(q_1, q_2)q_{d2} + G(q_1, q_2) - U). \tag{7.10}$$

Our aim is to implement an intelligent UR control [1] based on neural networks. Without loss of generality of the proposed approach, let's choose a two-layer NN (Figure 7.1). Let the hidden and output layers have H and m neurons, respectively (m is equal to the dimension of e_2). For the sake of simplicity, one supposes that only the sum of weighted signals (without nonlinear transformation) is realized in the neural network output layer. The input vector has N coordinates.

Let's define w_{ij} as the weight coefficient for the i-th input of the j-th neuron of the hidden layer. So, these coefficients compose the following matrix

$$w = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1N} \\ w_{21} & w_{22} & \dots & w_{2N} \\ \dots & \dots & \dots & \dots \\ w_{H1} & w_{H2} & \dots & w_{HN} \end{bmatrix}. \tag{7.11}$$

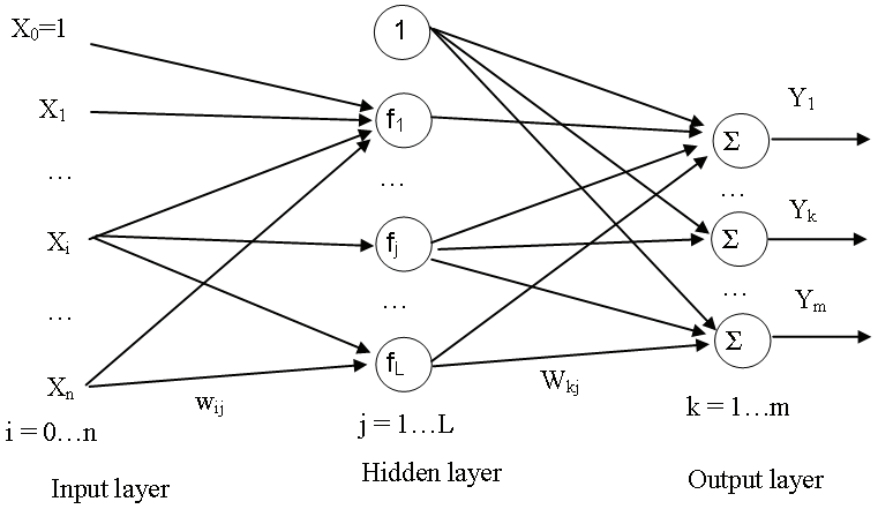

Figure 7.1 Neural network structure.

As a result of the nonlinear transformation $f\,(w, x)$, the hidden layer output vector can be written in the following form:

$$f(w,x) = \begin{bmatrix} f_1(w_1^T x) \\ \dots \\ f_H(w_H^T x) \end{bmatrix}, \tag{7.12}$$

where w_k denotes the k-th raw of matrix w and x is the NN input vector.

Analogously, let's introduce the matrix W whose element W_{li} denotes the transform (weight) coefficient from the i-th neuron of the hidden layer to the l-th neuron of the output layer.

Once the NN parameters are defined, the underwater robot control signal (NN output) is computed as follows:

$$U = y(W, w, x) = W \cdot f(w, x). \tag{7.13}$$

Substitution of this control into (7.10), allows us to get

$$\begin{aligned} \dot{Q} = e_2^T (D(q_1)\dot{q}_{d2} + B(q_1, q_2)q_{d2} + \\ + G(q_1, q_2) - W \cdot f(w, x)). \end{aligned} \tag{7.14}$$

To derive the NN learning algorithm, we apply the speed gradient method [4]. For this, we compute the partial derivatives of the time derivative of

function Q with respect to the adjustable NN parameters – matrices and W. Direct differentiation gives

$$\frac{\partial \dot{Q}}{\partial W} = -e_2 f^T(w, x). \tag{7.15}$$

It is easy to demonstrate that if we choose all activation functions in the usual form

$$f\ (x) = 1/(1 + e^{-\tau x}), \tag{7.16}$$

this implies the following property

$$\frac{\partial}{\partial w_{ij}} f_i(w_i^T x) = f_i(w_i^T x)[1 - f_i(w_i^T x)]x_j. \tag{7.17}$$

Let's introduce the following additional functions

$$\phi_i(w_i^T x) = f_i(w_i^T x)[1 - f_i(w_i^T x)] \tag{7.18}$$

and the matrix

$$\Phi(w, x) = diag(\phi_1(w_1^T x)...\phi_H(w_H^T x)). \tag{7.19}$$

Hence, direct calculation gives

$$\frac{\partial \dot{Q}}{\partial w} = -\Phi W^T e_2 x^T. \tag{7.20}$$

As a final stage, one can write the NN learning algorithm in the following form:

$$\begin{aligned} W^{(k+1)} &= W^{(k)} + \gamma e_2 f^T(w, x), \\ w^{(k+1)} &= w^{(k)} + \gamma \Phi W^T e_2 x^T, \end{aligned} \tag{7.21}$$

where γ is the learning step, k is the number of iterations.

The continuous form of this learning algorithm can be presented as

$$\begin{aligned} \dot{W} &= \gamma e_2 f^T(w.x), \\ \dot{w} &= \gamma \Phi W e_2 x^T(w.x). \end{aligned} \tag{7.22}$$

Such an integral law of the NN-regulator learning algorithm may cause unstable regimes in the control system, as it takes place in adaptive systems [4]. The following robust form of the same algorithm is also used:

$$\begin{aligned} \dot{W} &= \gamma e_2 f^T(w.x) - \alpha W, \\ \dot{w} &= \gamma \Phi W e_2 x^T(w.x) - \alpha w, \end{aligned} \tag{7.23}$$

where constant $\alpha > 0$.

Now, let's consider which components should be included in the NN input vector. The NN controller is oriented to compensate an influence of the appropriate matrix and vector functions, and thus, in the most common case, the NN input vector must be composed of q_1, q_2, e_2, q_{d2} and their time derivative.

The NN learning procedure leads to the reduction of function Q, and thus, in ideal conditions, the error e_2 converges to zero and the UR follows the desired trajectory

$$q_2(t) \rightarrow q_{d2}(t). \tag{7.24}$$

If the UR trajectory is given by $q_{d1}(t)$, one can choose

$$q_{d2}(t) = J^{-1}(q_1)(\dot{q}_{d1}(t) + k(q_{d1}(t) - q_1(t))), \tag{7.25}$$

where k is a positive constant. From the kinematics Equation (7.1), it follows that

$$\dot{q}_1(t) \rightarrow \dot{q}_{d1}(t) + k(q_{d1}(t) - q_1(t)) \tag{7.26}$$

and

$$\dot{e}_1(t) + ke_1(t) \rightarrow 0, \tag{7.27}$$

where

$$e_1(t) = q_{d1}(t) - q_1(t). \tag{7.28}$$

Hence, the UR follows to the planned trajectory $q_{d1}(t)$.

7.4 Simulation Results of the Intelligent NN Controller

In order to check the effectiveness of the proposed approach, different computer simulations have been carried out. The UR model parameters were taken from [6]. The UR parameters are the following:

$$D = D_{RB} + D_A,$$

where the inertia matrix of the UR rigid body is

$$D_{RB} = \begin{bmatrix} 1000 & 0 & 200 \\ 0 & 1000 & 0 \\ 200 & 0 & 11000 \end{bmatrix},$$

and the inertia matrix of the hydrodynamic added mass is

$$D_A = \begin{bmatrix} 1000 & 0 & 100 \\ 0 & 1100 & 80 \\ 100 & 80 & 9000 \end{bmatrix}.$$

Matrices *B* and *G* are

$$B = \begin{bmatrix} 210 & 20 & 30 \\ 25 & 200 & 70 \\ 15 & 33 & 150 \end{bmatrix},$$

$$G = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T.$$

Vector q_2 consists of the following components (linear and angular UR velocities):

$$q_2 = \begin{bmatrix} v_x & v_z & \omega_y \end{bmatrix}^T. \tag{7.29}$$

The NN input is composed by q_2 and e_2. The NN output (control forces and torque) is the vector

$$U = \begin{bmatrix} F_x & F_z & M_y \end{bmatrix}^T. \tag{7.30}$$

For the NN controller containing 10 neurons in the hidden layer, the simulation results are given on Figures 7.2 – 7.9.

In the considered numerical experiments, the desired trajectory was taken as follows:

$$\begin{cases} v_{xd} = 0.75m/\sec, \\ v_{zd} = 0.5m/\sec, \\ \omega_{yd} = -0.15rad/\sec, \end{cases} \quad 0 \le t \le 250\sec,$$

$$\begin{cases} v_{xd} = 0.5m/\sec, \\ v_{zd} = 0.75m/\sec, \\ \omega_{yd} = 0.15rad/\sec. \end{cases} \quad 250 \le t \le 500\sec$$

7.5 Modification of NN Control

In previous sections, a NN control was designed. Practically speaking, the synthesis procedure of the NN regulator does not use any information of the mathematical model of the controlled object. As one can see, differential

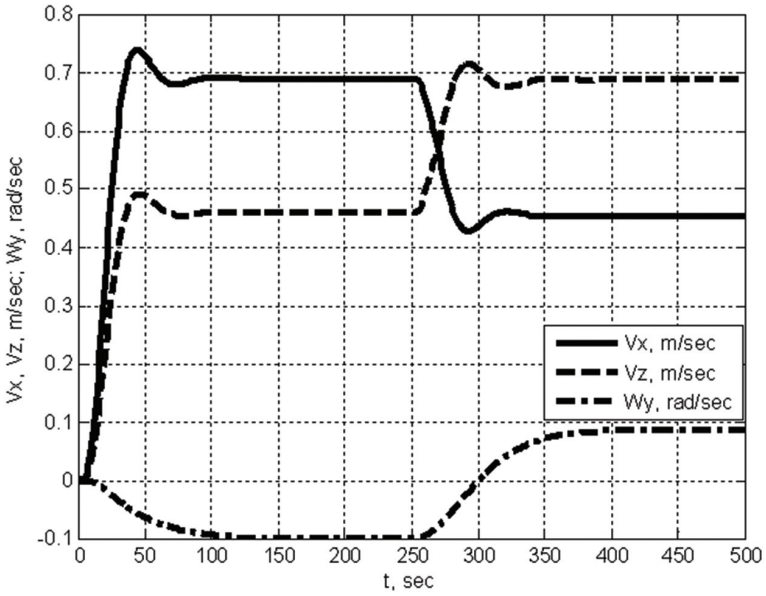

Figure 7.2 Transient processes in NN control system ($\alpha = 0.01$, $\gamma = 250$).

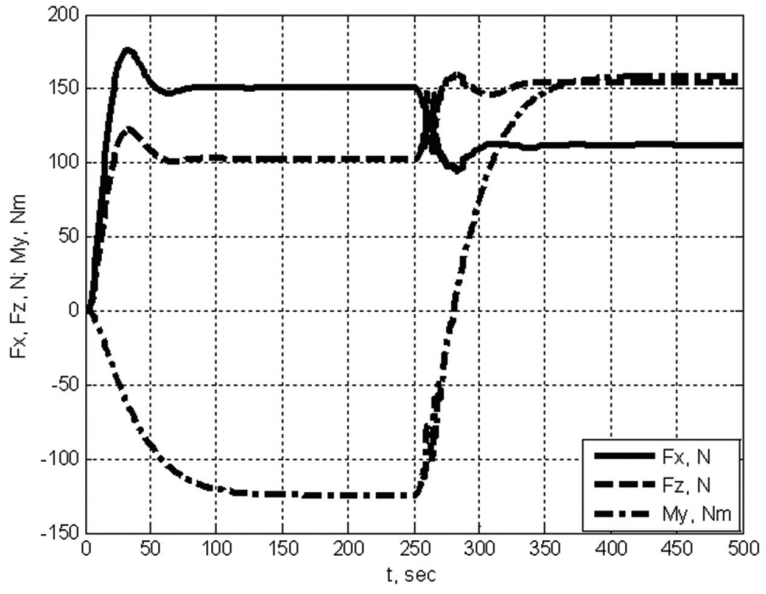

Figure 7.3 Forces and Torque in NN control system ($\alpha = 0.01$, $\gamma = 250$).

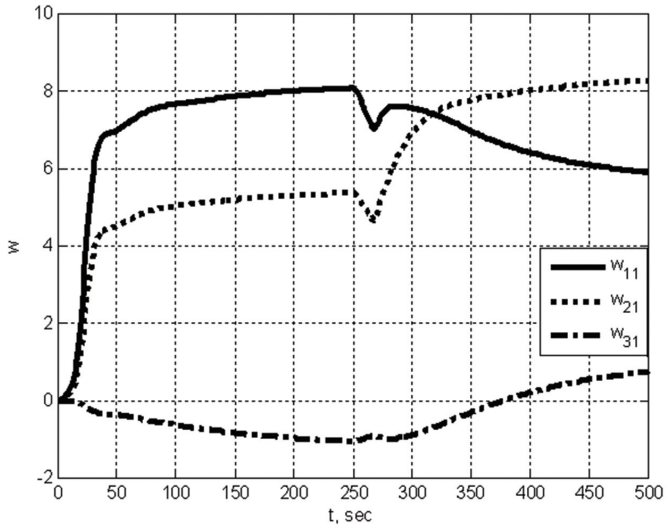

Figure 7.4 Examples of hidden layer weight coefficients evolution ($\alpha = 0.01, \gamma = 250$).

equations describing the underwater robot dynamics have a particular structure which can be taken into account for solving the synthesis problem of the control system.

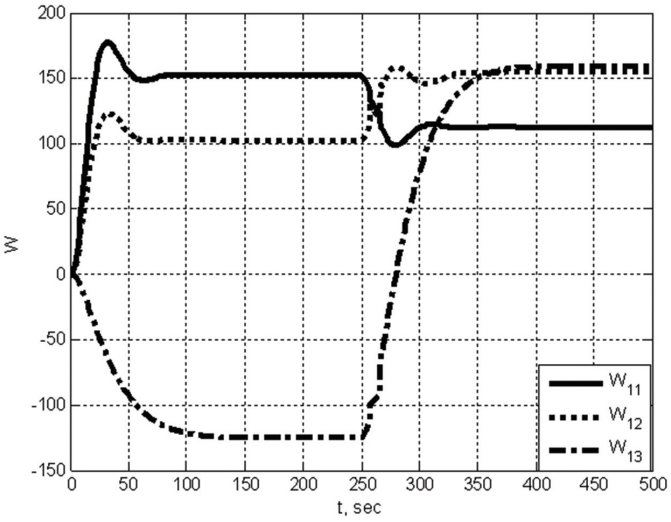

Figure 7.5 Examples of output layer weight coefficients evolution ($\alpha = 0.01$, $\gamma = 250$).

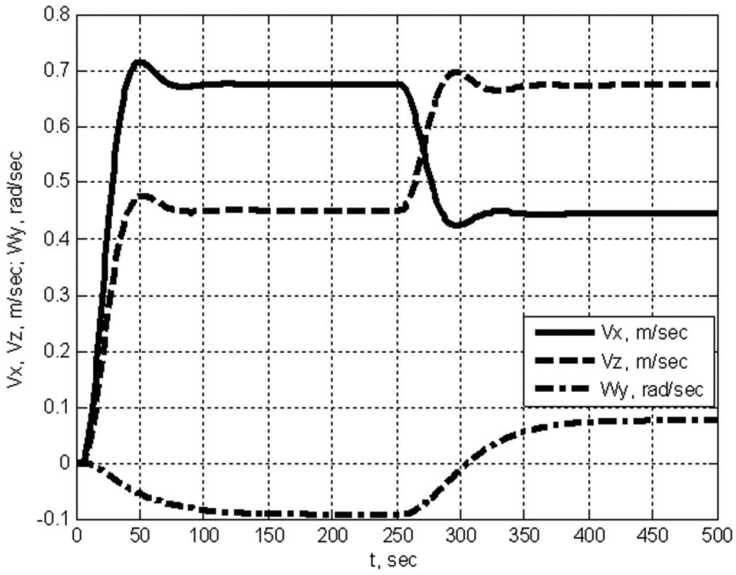

Figure 7.6 Transient processes in NN control system ($\alpha = 0.01$, $\gamma = 200$).

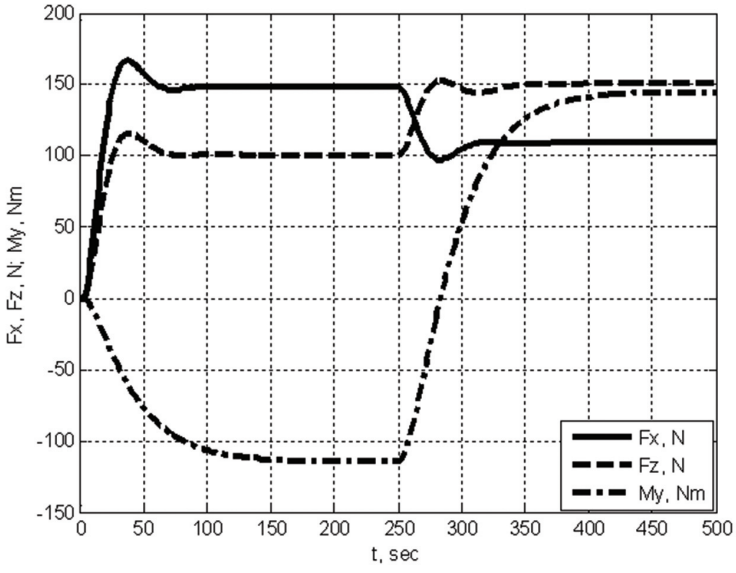

Figure 7.7 Forces ant Torque in NN control system ($\alpha = 0.01$, $\gamma = 200$).

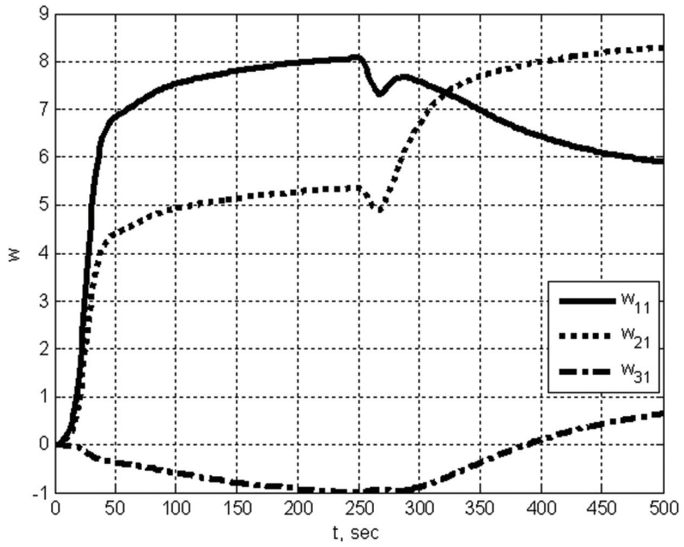

Figure 7.8 Examples of hidden layer weight coefficients evolution (α = 0.01, γ = 200).

There exist different ways to solve it. One of the possible approaches is derived below:

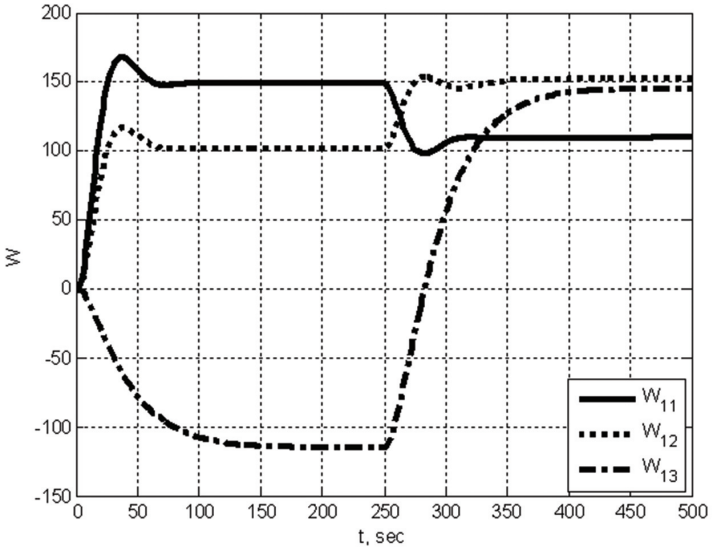

Figure 7.9 Examples of output layer weight coefficients evolution (α = 0.01, γ = 200).

As mentioned before, the parameters of underwater robots, such as added masses, moments of inertia, coefficients of viscous friction etc, are not all exactly known because of the complex hydrodynamic nature of the robot movement in the water environment.

Let's suppose that a set of nominal UR parameters can be estimated. Hence, it is possible to get appropriate nominal matrices $D_0(q_1)$, $B_0(q_1, q_2)$ and $G_0(q_1, q_2)$ in Equation (7.2). Let's denote the deviations of the real matrices from the nominal ones as $\Delta D(q_1)$, $\Delta B(q_1, q_2)$ and $\Delta G(q_1, q_2)$, respectively. So, the following takes place:

$$\begin{aligned} D(q_1) &= D_0(q_1) + \Delta D(q_1), \\ B(q_1, q_2) &= B_0(q_1, q_2) + \Delta B(q_1, q_2), \\ G(q_1, q_2) &= G_0(q_1, q_2) + \Delta G(q_1, q_2). \end{aligned} \tag{7.31}$$

Inserting expressions (7.29) into Equation (7.10) gives

$$\begin{aligned} \dot{Q} = e_2^T (&D_0(q_1)\dot{q}_{d2} + B_0(q_1, q_2)q_{d2} + G_0(q_1, q_2) + \\ &+ \Delta D(q_1)\dot{q}_{d2} + \Delta B(q_1, q_2)q_{d2} + \Delta G(q_1, q_2) - U). \end{aligned} \tag{7.32}$$

Now let's choose the control law in the form:

$$U = U_0 + U_{NN}, \tag{7.33}$$

where

$$U_0 = D_0(q_1)\dot{q}_{d2} + B_0(q_1, q_2)q_{d2} + G_0(q_1, q_2) + \Gamma e_2, \tag{7.34}$$

is the nominal control associated with the known part of the robot dynamics (matrix $\Gamma > 0$ is positively definite) and U_{NN} is the neural network control to compensate the uncertainty. The scheme of the proposed NN control system for an underwater robot is given on Figure 7.10.

If the robot dynamics can be exactly determined (and uncertainty does not take place), the nominal control (7.34) fully compensates undesirable terms in (7.32) (U_{NN}can be taken as equal to zero) and one has

$$\dot{Q} = -e_2^T \Gamma e_2 < 0. \tag{7.35}$$

Thus, functions $Q(t)$ and $e_2(t)$ converge to zero for $t \to \infty$.

In the general case, as follows from (7.32) – (7.34), one has

$$\dot{Q} = e_2^T (\Delta D(q_1)\dot{q}_{d2} + \Delta B(q_1, q_2)q_{d2} + \Delta G(q_1, q_2) - U_{NN}). \tag{7.36}$$

As one can expect, the use of the nominal component of the control facilitates the implementation of the proper NN control.

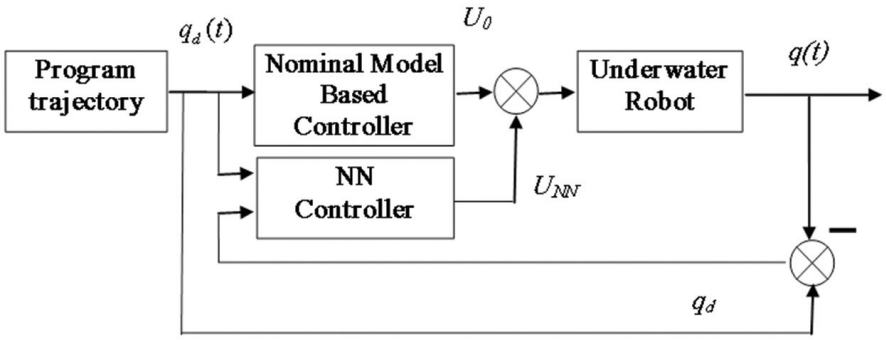

Figure 7.10 Scheme of the NN control system.

Further steps of the NN controller learning algorithm can be done practically in the same manner as above (see Equation (7.15, 20 and 21)).

In order to check the derived NN control, mathematical simulations of the UR control system were carried out. The nominal matrices $D_0(q_1)$, $B_0(q_1, q_2)$ and $G_0(q_1, q_2)$ were taken as follows:

$$D_0 = D_{RB0} + D_{A0},$$

$$D_{RB0} = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 11000 \end{bmatrix}, D_{A0} = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1100 & 0 \\ 0 & 0 & 9000 \end{bmatrix},$$

$$B_0 = \begin{bmatrix} 210 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 150 \end{bmatrix},$$

$$G_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}.$$

and matrix $\Gamma = diag\,[0.02,\ 0.02,\ 0.02]$.

Note that the matrices D_0, B_0 of the nominal dynamics model contain only diagonal elements which are not equal to zero. This means that the nominal model is simplified and does not take into account an interaction between different control channels (of linear and angular velocities). The absence of these terms in the nominal dynamics results in partial parametric and structural uncertainty.

Figures 7.11 – 7.18 show the transient processes and control signals (forces and torque) in the designed system with a modified NN regulator. The experimental results demonstrated that the robot coordinates converge to

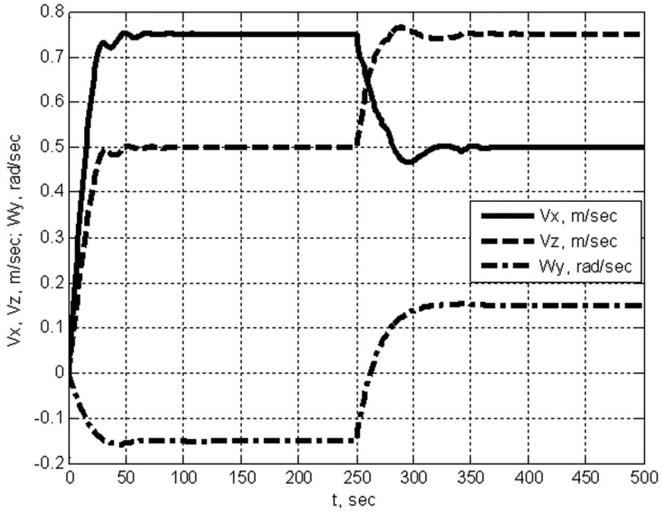

Figure 7.11 Transient processes with modified NN-control ($\alpha = 0$, $\gamma = 200$).

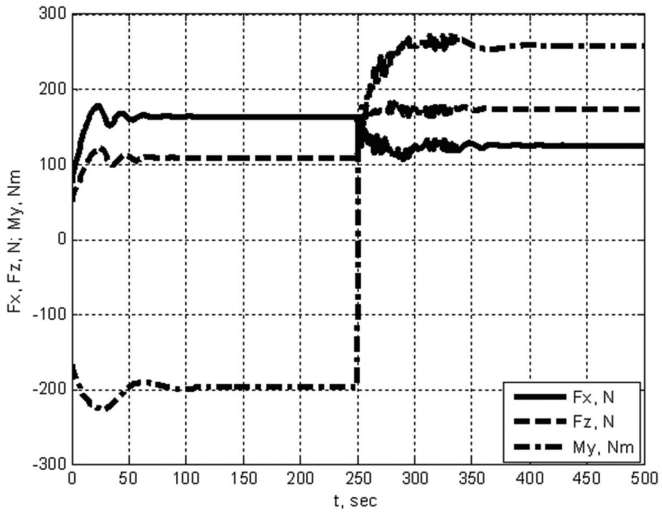

Figure 7.12 Forces and torque with modified NN control ($\alpha = 0$, $\gamma = 200$).

the desired trajectories. In comparison with the conventional multilayer NN applications, the weight coefficients of the proposed NN controller are varying simultaneously with the control processes.

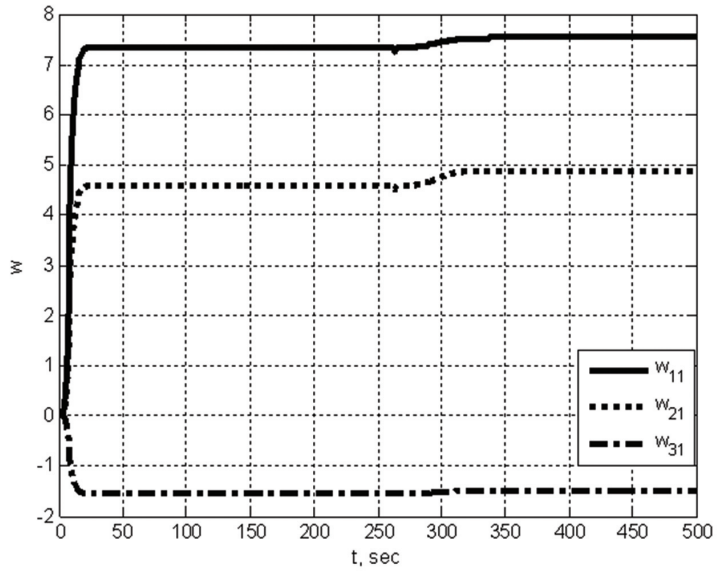

Figure 7.13 Examples of hidden layer weight coefficients evolution ($\alpha = 0$, $\gamma = 200$).

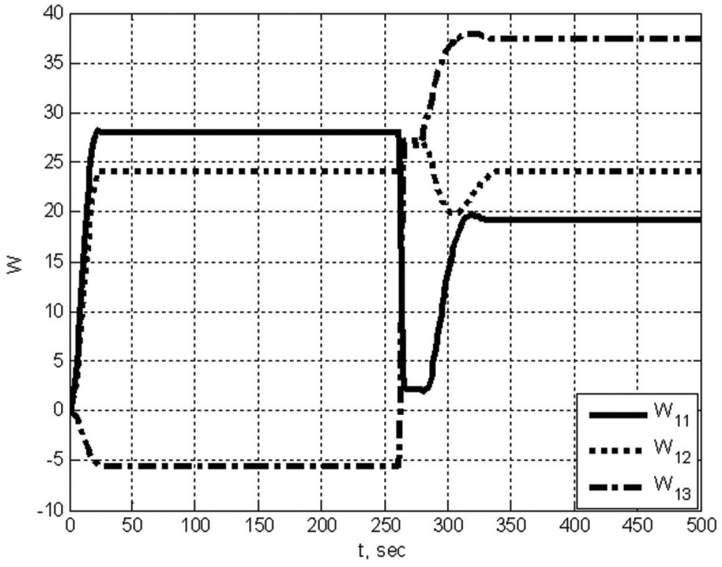

Figure 7.14 Examples of output layer weight coefficients evolution ($\alpha = 0$, $\gamma = 200$).

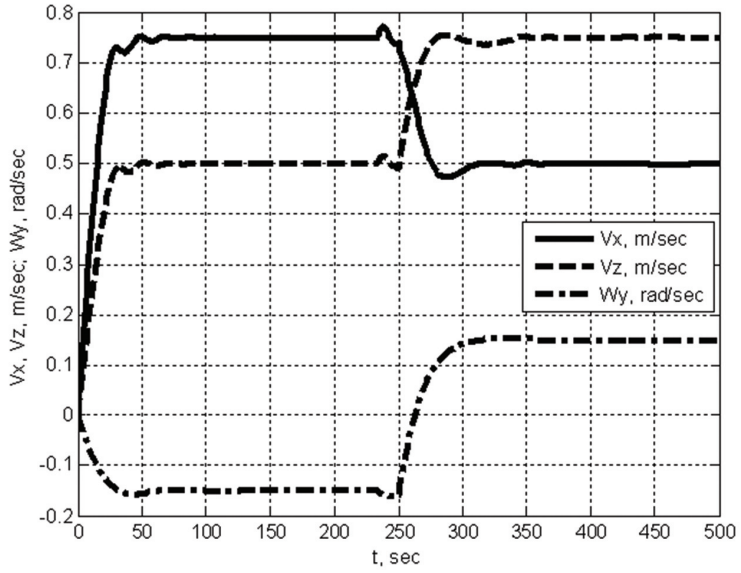

Figure 7.15 Transient processes with modified NN control (α =0.001, γ = 200).

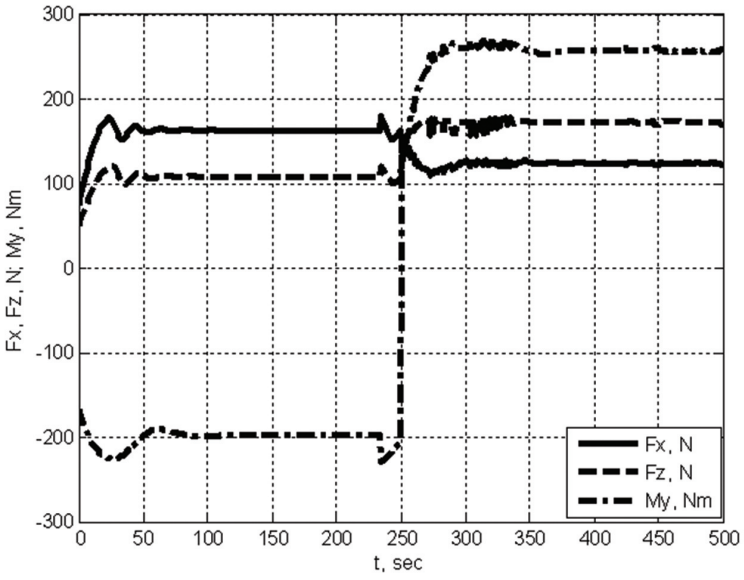

Figure 7.16 Forces and Torque with modified NN control (α = 0.001, γ = 200).

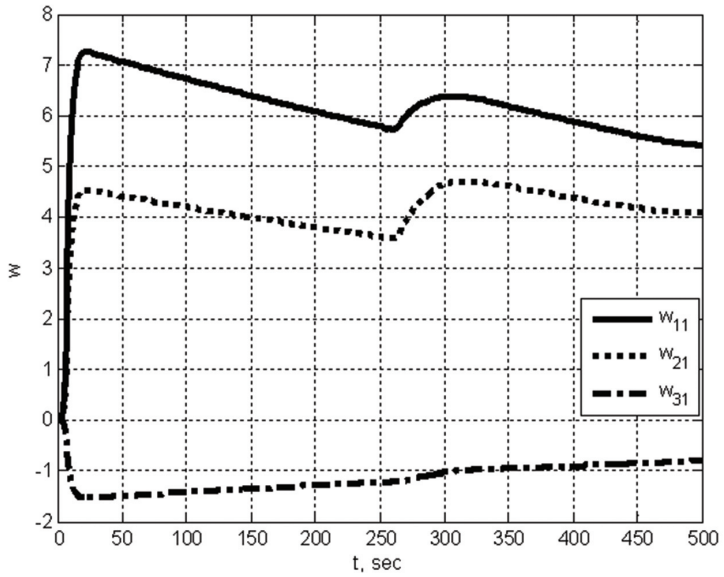

Figure 7.17 Examples of hidden layer weight coefficients evolution (α =0.001, γ =200).

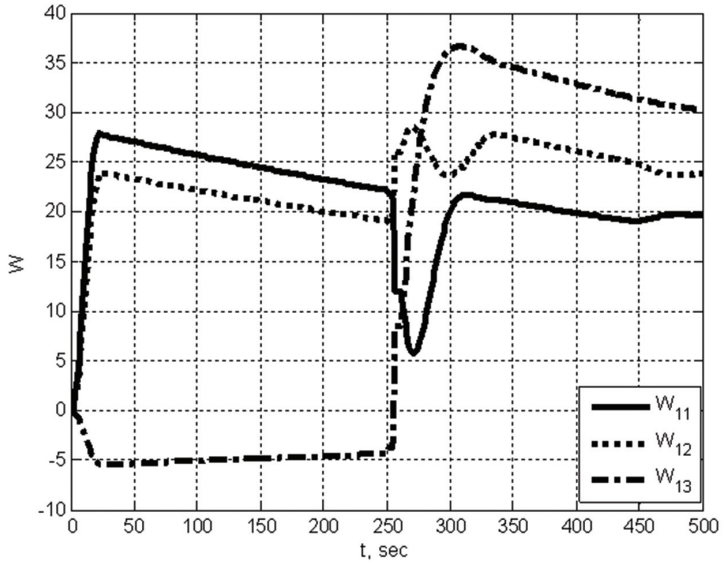

Figure 7.18 Examples of output layer weight coefficients evolution (α =0.001, γ = 200).

7.6 Conclusions

An approach on how to design an intelligent NN controller for underwater robots and how to derive its learning algorithm on the basis of a speed gradient method is proposed and studied in this chapter. The numerical experiments have shown that high-quality processes can be achieved with the proposed intelligent NN control. In the study case of producing an UR control system, the NN learning procedure allows to overcome the parameter and partial structural uncertainty of the dynamical object. The combination of the neural network approach with the proposed control, designed using the nominal model of the underwater robot dynamics, allows to simplify the control system implementation and to improve the quality of the transient processes.

Acknowledgement

The work of A.Dyda and D.Oskin was supported by Ministry of Science and Education of Russian Federation, the State Contract No 02G25.31.0025.

References

[1] A. A. Dyda, 'Adaptive and neural network control for complex dynamical objects', - Vladivostok, Dalnauka. 2007. pp. 149 (in Russian).

[2] A. A. Dyda, D. A. Oskin, 'Neural network control system for underwater robots.' IFAC conference on Control Application in Marine Systems "CAMS-2004", - Ancona, Italy, 2004, pp. 427–432.

[3] T. I. Fossen, 'Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles', Marine Cybernetics AS, Trodheim, Norway, 2002.

[4] A. A. Fradkov, 'Adaptive control in large-scale systems', - M.: Nauka., 1990, (in Russian).

[5] A. A. Narendra, K. Parthasaraty, 'Identification and control of dynamical systems using neural networks', IEEE Identification and Control of Dynamical System, Vol.1. No 1. 20, 1990, pp. 1475–1483.

[6] A. Ross, T. Fossen and A. Johansen, 'Identification of underwater vehicle hydrodynamic coefficients using free decay tests', Preprints of Int. Conf. CAMS-2004, Ancona, Italy, 2004. pp. 363–368.

[7] R. Sutton and A. A. Craven, 'An on-line intelligent multi-input multi-output autopilot design study', Journal of Engineering for the Maritime Environment, vol. 216. No. M2, 2002, pp. 117–131.
[8] J. Yuh, 'Modeling and control of underwater robotic vehicles', Systems, Man and Cybernetics, IEEE Transactions on, vol. 20, no. 6, pp. 1475–1483, Nov/Dec 1990. doi: 10.1109/21.61218