Circuits Systems Signal Process
Vol. 14, No. 2, 1995, Pp. 213–236

An Improved Scheme for Direct Adaptive Control of Dynamical Systems Using Backpropagation Neural Networks*

K. P. Venugopal,[1] *R. Sudhakar,*[2] *and A. S. Pandya*[3]

Abstract. This paper presents an improved direct control architecture for the on-line learning control of dynamical systems using backpropagation neural networks. The proposed architecture is compared with the other direct control schemes. In this scheme the neural network interconnection strengths are updated based on the output error of the dynamical system directly, rather than using a transformed version of the error employed in other schemes. The ill effects of the controlled dynamics on the on-line updating of the network weights are moderated by including a compensating gain layer. An error feedback is introduced to improve the dynamic response of the control system. Simulation studies are performed using the nonlinear dynamics of an underwater vehicle and the promising results support the effectiveness of the proposed scheme.

1. Introduction

The recent interest in highly parallel networks of simple processing elements (neurons) had led to the wide use of such neural networks (NN) for the learning control of dynamical systems [3]–[7], [9]–[11], [14],[16],[20]. The capabilities of such networks for efficient generalization and adaptation make them excellent candidates for the learning control of linear as well as nonlinear dynamical systems. The objective of an NN controller is to generate the correct control signal to drive the dynamics from the initial state to a desired final state through an optimum state trajectory. The tracking performance and the ease of implementation of a neural network controller largely depends upon the learning algorithm chosen as well as the control architecture used. The two generally used control architectures

* Received March 22, 1993; revised September 20, 1993.

[1] Currently with the Medical Image Processing Group, University of Pennsylvania, Philadelphia, PA 19104.

[2] Department of Electrical Engineering, Florida Atlantic University, Boca Raton, FL 33431.

[3] Center for Advanced Marine Systems, and Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431.

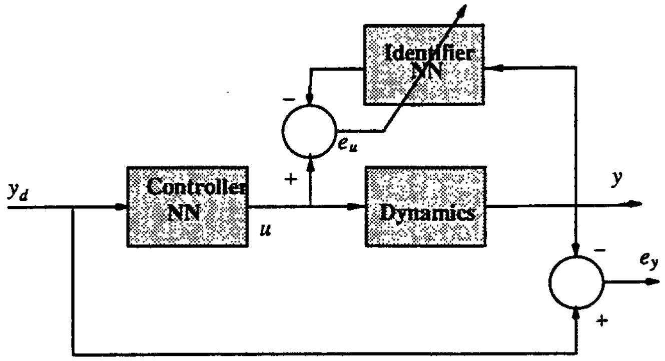

Figure 1a. Indirect control scheme.

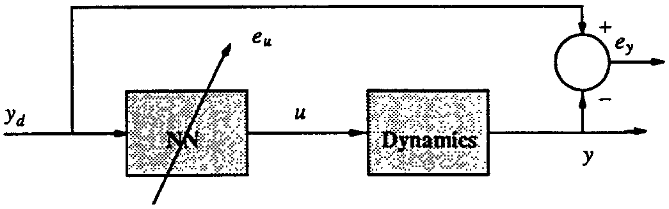

Figure 1b. Direct control scheme. The error e_u corresponds to the error at the output of the neural network.

are the direct control scheme and the indirect control scheme. In most of these implementations, backpropagation is used as the learning algorithm.

In the indirect control scheme, the parameters of the dynamical system are identified at each instant; these parameters are then used to estimate the controller parameters (see Figure 1a). Thus there is an explicit identification process in this approach [9], [8]. The disadvantage of such a scheme is that the identification and control are solely based on the error e_u, and hence minimization of the output error e_y cannot be guaranteed [11]. In the direct control scheme, the controller parameters are directly adjusted to reduce the output error (Figure 1b). Hence it is simpler to implement such a scheme [8],[9]. It may be noted that even when the dynamics is linear and time invariant, the above approaches result in overall nonlinear systems [8],[9]. One of the problems in having a direct control scheme using neural networks is that the error at the output of the neural network (network error e_u), which is needed for updating the network weights, is not directly available (see Figure 1b). Only the system error e_y can be measured at the output of the dynamics, and it does not represent the error at the NN output. Three different methods have been proposed in the literature to derive the network error from the system error measurements. In the first method [11], the dynamics is treated as an unmodificable layer of the NN, and the error is backpropagated through the dynamics. In the method proposed by Chen et al. [1], the system error is transformed into the network error using an inverse transfer matrix of the dynamics. The third method [3] uses a fixed feedback gain stage to generate a transformed error signal that is employed for updating the neural networks. In the first two methods, the differential gain of the dynamics (forward Jacobian) directly affects the network updating. These methods suffer from the fact that an on-line numerical estimation of the Jacobian may result in sharp changes in the transformed error and hence in the network weights. On the other hand, the third method may be able to provide robust results only over a particular range, as the feedback gain is fixed [1],[3].

In this paper, after a brief review of the above three implementations of the direct control scheme using the backpropagation algorithm, we propose a modified architecture. The major difference in the proposed architecture is that the network interconnection strengths are updated using the system output error rather than the transformed error used in the other schemes. The ill effects due to the numerical estimation of the Jacobian of the controlled dynamics are moderated by adding an extra gain layer between the neural network controller and the system. The gain layer is a single layer NN with linear transformation. The gain layer NN is initialized to the approximate steady state inverse gain of the dynamics and is adapted on-line. An error feedback, similar to the one in Kawato's scheme [3] is introduced to improve the dynamic response of the control system. Results of simulation studies performed on the nonlinear dynamics of an underwater vehicle show that the presented schemes give promising results. In the following sections we describe the schemes and the simulation results. In this study, we first develop and analyze the schemes for Single Input Single Output (SISO) dynamics for ease of explanation and comparison. The schemes are extended to deal with Multiple Input Multiple Ouput (MIMO) dynamics. Simulation results are provided for the control of both SISO and MIMO dynamics. Also, in the subsequent discussions, we use the words controller, NN, and network interchangeably, to refer to the neural network controller.

2. Direct control schemes

Consider a time-varying dynamical system represented by a set of nonlinear differential equations, $\dot{x} = f(x, u, t)$, $y = h(x, t)$, where x denotes the state vector, y the system output, and u the input. If we represent the input to the dynamics u as a nonlinear function of x and t, i.e., $u = \phi(x, t)$, then $\dot{x} = f[x, \phi(x, t), t]$ represents the closed loop system dynamics. The objective of an on-line learning controller is to generate the control signal to drive the dynamics to a state x^* at each instant in time so that the error $e_y(t)$ between the desired output $y_d(t)$ and the actual output $y(t)$ asymptotically tends to zero.

In an NN based on-line learning control scheme the objective of the neural network is to generate the control input u by mapping the nonlinear function $\phi(x, t)$. As indicated previously, to use the backpropagation learning algorithm, the error at the output of the network has to be estimated in order to update the neural network interconnection strengths.

2.1. *The network error and the system error*

Figure 1b shows the basic direct control scheme. The NN learns the inverse characteristics of the dynamics implicitly and generates the correct control signal u to drive the system state y to the desired state y_d. Defining the squared error at the system output as the objective function to be minimized by the controller, we have:

$$E_y = \frac{1}{2}(y_d - y)^2. \tag{1}$$

If $(y_d - y)$ is defined as e_y, the gradient of the error with respect to the network interconnection strength w is:

$$\frac{\partial E_y}{\partial w} = -e_y \frac{\partial y}{\partial w} \tag{2}$$

$$= -e_y \frac{\partial y}{\partial u} \frac{\partial u}{\partial w}. \tag{3}$$

The above equations can be generalized to the case when w is a vector. The term $\frac{\partial y}{\partial u}$ corresponds to the forward gain of the dynamics and is an important factor in determining the stability of the overall system [19]. The term $e_y \frac{\partial y}{\partial u}$ represents the transformation of the total error e_y with a transformation factor $\frac{\partial y}{\partial u}$. If we define $J(u) = \frac{\partial y}{\partial u}$, equation (3) can be rewritten as

$$\frac{\partial E_y}{\partial w} = -e_y J(u) \frac{\partial u}{\partial w}. \tag{4}$$

2.2. *Backpropagating the error through the dynamics*

If we consider the system dynamics as an additional layer of the NN controller, having interconnection strength $J(u)$ with no updating, equation (4) represents the backpropagation of the system error e_y through the dynamics. This approach, originally proposed by Psaltis et al. [11] and termed the "specialized learning scheme," is an effective way for the direct control of dynamical systems using neural networks. It is used in a number of later studies in different forms [4],[16],[20]. We review the network updating equations here for comparison with the other schemes discussed in this paper.

Following the notations in [12, pp. 327–330], let us define Δw_{ji}^p as the change in weight w_{ji} connecting neuron i in the $(p-1)$th layer to the neuron j in the pth layer. Let δ_j^p be the derivative of the error at the layer p, say E_p, with respect to the summed input to the sigmoid transformation (net_j^p) at neuron j,and o_j^p as the output of the neuron. Then the updating equations for any layer can be shown to be [12]:

$$w_{ji}^p(n+1) = w_{ji}^p(n) + \Delta w_{ji}^p(n) \tag{5a}$$

where

$$\Delta w_{ji}^p = \eta \delta_j^p o_i^p \tag{5b}$$

and η is the learning rate of the NN.

Using equation (4), we can get δ_j^p for the output layer of the network as

$$\delta_j^p = J(u) e_y f_j'(net_j^p) \tag{6a}$$

and for any other layer as

$$\delta_i^p = f_i'(net_i^p) \sum_j \delta_j^p w_{ji} \tag{6b}$$

where f_i' denotes the derivative of the sigmoidal transformation function f_i which is given by

$$f_i(net_i^p) = \frac{[1 - \exp(-net_i^p)]}{[1 + \exp(-net_i^p)]}. \tag{7}$$

The differential gain $J(u)$ (or instantaneous forward Jacobian) can be estimated on-line from the changes in y and u over two consecutive time instants. Alternately, the dynamics can be modeled using a neural network and can be used for transforming the error e_y to e_u, the equivalent error at the NN output (see Figure 1b) [6]. Troudet et al. discusses such a scheme for flight control application in [16].

One of the problems in treating the dynamics as a nonmodifiable layer is that the convergence of the NN may be slow [1]. From equation (6a), it can be seen that the differential gain of the dynamics directly affects the network updating. An on-line numerical estimation procedure using the ratio of the incremental changes in the output and input of the system dynamics may generate large variations in the value of $J(u)$, causing correspondingly large changes in weights. This problem is more evident when the reference command signals are rapidly varying. Also, under steady state conditions when the change in inputs and outputs are very small, a numerically estimated Jacobian $J(u)$ will be inaccurate. The inverse transfer matrix scheme suggested by Chen et al. [1] is a more suitable method under such conditions.

2.3. The inverse transfer matrix scheme

In the inverse transfer matrix scheme (Figure 2), the error function to be minimized is chosen as

$$E_{K_y} = \frac{1}{2}(K_c e_y)^2 \tag{8}$$

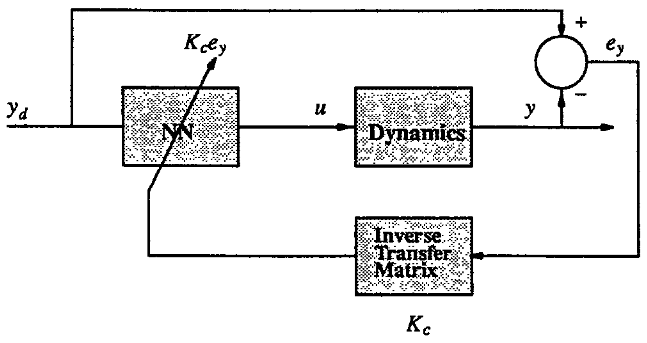

Figure 2. Inverse transfer matrix scheme.

where K_c represents the inverse transfer relationship of the dynamics. The network updating is done by estimating the derivative of the error function with respect to the weights, i.e.,

$$\frac{\partial E_{K_y}}{\partial w} = -K_c^2 e_y \frac{\partial y}{\partial w}. \tag{9}$$

As before $J(u) = \frac{\partial y}{\partial u}$ and equation (9) reduces to:

$$\frac{\partial E_{K_y}}{\partial w} = -K_c^2 e_y J(u) \frac{\partial u}{\partial w}. \tag{10}$$

When $K_c \simeq [J(u)]^{-1}$, the scheme proposed in [1] is obtained; E_{K_y} corresponds to the squared error at the NN output. Then

$$\frac{\partial E_{K_y}}{\partial w} = -K_c e_y \frac{\partial u}{\partial w}. \tag{11}$$

The updating rules in this scheme remain the same as in Psaltis' scheme, except for the output layer weights. For the output layer weights, equation (6a) is modified as

$$\delta_j^p = K_c e_y f_j'(net_j^p). \tag{12}$$

The important difference between the specialized learning scheme and the inverse transfer matrix scheme is that in the first method the system error e_y is minimized, whereas in the second method the error estimated at the output of the NN, e_u, is minimized. Comparing equations (6a) and (12), it can be seen that the differential gain of the dynamics affects the network updating indifferent ways in the above schemes. In the case of the inverse transfer matrix scheme [1], if the inverse differential gain K_c is computed off-line, the convergence may be faster because the dynamics is not treated as an additional layer of the NN. However, the numerical problems associated with the on-line computation of K_c can cause large changes in the network weights. Also, for control signals of constant values ($\partial u \simeq 0$), the on-line estimated values of the inverse differential gain may be inaccurate and unreliable.

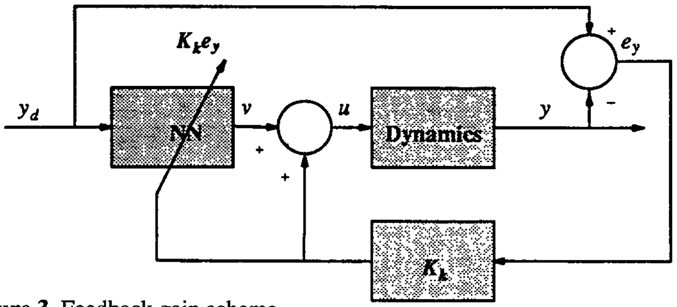

Figure 3. Feedback gain scheme.

2.4. The feedback gain scheme

Kawato et al. [3] suggested an error feedback scheme for the direct control of dynamical systems, motivated by the voluntary movements in biological systems (Figure 3). The feedback gain K_k is estimated off-line and is used for transforming the error for updating the network weights as well as for generating the control signal u. From Figure 3, we obtain the following expressions for the control signal u:

$$u = K_k e_y + v \tag{13}$$

where $e_y = y_d - y$, as before, and v is the NN output. In terms of the Jacobian of the dynamics $J(u)$, the system response y can be expressed as:

$$y = [v + K_k(y_d - y)]J(u). \tag{14}$$

Expanding equation (14) and rewriting:

$$y = \frac{J(u)v}{1 + J(u)K_k} + \frac{K_k J(u) y_d}{1 + J(u)K_k}. \tag{15}$$

The scheme in [3] corresponds to the approximation $K_k \simeq [J(u)]^{-1}$. Then equation (15) reduces to:

$$y = \frac{J(u)v}{2} + \frac{y_d}{2}. \tag{16}$$

Note that the command signal y_d is directly influencing the system response because of the error feedback.

The error function to be minimized in this scheme is the same as that of equation (8), and the network updating equations can be derived as before. From equation (14) we get

$$\frac{\partial y}{\partial w} = J(u)\left[\frac{\partial v}{\partial w} - K_k \frac{\partial y}{\partial w}\right] \tag{17}$$

i.e.,

$$\frac{\partial y}{\partial w} = \frac{J(u)\, \partial v/\partial w}{1 + J(u)K_k}. \tag{18}$$

When $K_k \simeq [J(u)]^{-1}$ in equation (18) and with E_{k_y} defined by be $\frac{1}{2}(K_k e_y)^2$, we obtain

$$\frac{\partial E_{k_y}}{\partial w} = -K_k \frac{e_y}{2} \frac{\partial v}{\partial w}. \tag{19}$$

Comparing equations (11) and (19) and using equation (6a), we obtain the weight updating equation for the output layer as:

$$\delta_j^p = K_k \frac{e_y}{2} f_j'(net_j^p). \tag{20}$$

Comparing equations (20) and (12), it can be seen that the feedback error scheme is equivalent to the inverse transfer matrix scheme when $J^{-1}(u) = \frac{K_k}{2}$. With off-line estimation of the Jacobian, both methods provide similar results for SISO dynamics.

It is to be noted that for good tracking performance over a wide range of nonlinear time-varying dynamics, sophisticated estimation of K_k is needed. An important aspect of this scheme is the increased speed of learning due to the error feedback. The network error directly affects the control input resulting in faster learning, whereas in the other schemes, the error is reflected only through the network. Note that the network learning rate is usually kept very small when using the backpropagation algorithm for control applications, to avoid the possibility of divergence [9],[19]. Thus when the error is reflected only through the NN, the convergence may be slow.

3. The gain layers schemes

In this section, we propose two different architectures for the direct control of dynamical systems using neural networks (Figures 4 and 5). The schemes use the system error e_y directly for updating the weights. The ill effects due to the numerical estimation of the differential gain of the dynamics are compensated by introducing a gain layer between the NN controller and the system. The gain layer is a separate single layer NN with linear transformations. The controller NN weights are updated at each iteration, according to the backpropagation algorithm using the error e_y. A rough estimate of the inverse of the steady state gain of the dynamics is used as the initial value of the gain layer NN and its weights are also updated at each iteration using the gradient descent approach.

The objective function to be minimized is chosen as $E_y = \frac{1}{2}(y_d - y)^2$. The NN weights can be updated using the total error e_y, provided we have $E_y \simeq E_v$, where E_v is the squared error at the output of the NN (see Figure 4). This condition is achieved when

$$\frac{\partial u}{\partial v} \simeq \frac{\partial u}{\partial y} \tag{21}$$

or

$$g \simeq [J(u)]^{-1}. \tag{22}$$

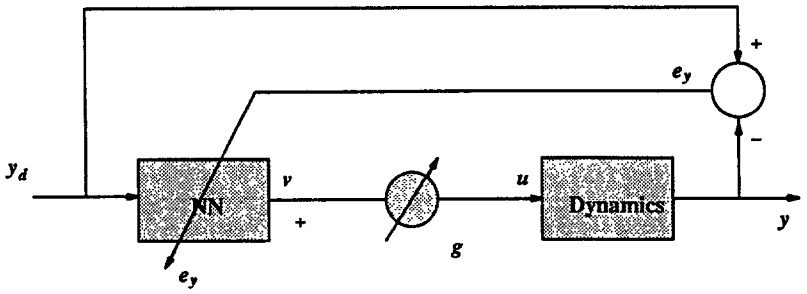

Figure 4. Gain layer scheme.

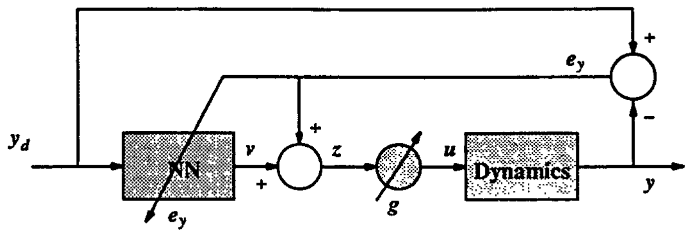

Figure 5. Gain layer scheme with error feedback.

In practice, the gain layer NN is determined through an adaptation process and will serve as an approximation of the inverse Jacobian of the dynamics. The gain layer has a role similar to the feedback gain constant K_k of Kawato's scheme and the inverse transfer matrix K_c of Chen et al. The weight g in the SISO case is a scalar value; it will be a matrix in the case of MIMO systems.

The NN updating equations are derived as in equations (5) and (6) and the output layer updating rule becomes

$$\delta_j^p = e_y f_j'(net_j^p). \tag{23}$$

In the following section, we outline the procedure for determining the gain layer weights. For simplicity, we first consider a SISO dynamics where g is a scalar value.

3.1. *Updating the gain layer*

Let $g(k)$ be the gain layer weight at the instant k. The adaptation rule for $g(k)$ can be written as

$$g(k) = g(k-1) + \Delta g(k) \tag{24}$$

where $\Delta g(k)$ is the change in $g(k)$ at instant k. Using the gradient descent rule to minimize the squared error E_y (see equation (1)), we obtain

$$\Delta g(k) = -a\frac{\partial E_y(k)}{\partial g(k)} \tag{25}$$

where α is the updating rate for $g(k)$. Rewriting the above equation using the chain rule and dropping the index for brevity,

$$\frac{\partial E_y}{\partial g} = \frac{\partial E_y}{\partial u}\frac{\partial u}{\partial g} \tag{26}$$

$$\frac{\partial E_y}{\partial u} = -(y_d - y)\frac{\partial y}{\partial u}. \tag{27}$$

Assuming that v is a weak function of g and ignoring the implicit effect of g on v through network feedback, we make the approximation that $\frac{\partial u}{\partial g} = v$. Thus equation (24) reduces to:

$$g(k) = g(k-1) + a(y_d - y)v\frac{\partial y}{\partial u}. \tag{28}$$

3.2. *Effect of the error feedback*

Equation (16) indicates that the error feedback helps to improve the system response. We introduce an error feedback similar to the scheme by Kawato and Figure 5 shows the modified scheme.

The input to the gain g is:

$$z = v + e_y = v + (y_d - y). \tag{29}$$

Then, the control input to the dynamics is given by

$$u = zg. \tag{30}$$

As before, defining

$$y = J(u)u \tag{31}$$

$$= J(u)g[v + (y_d - y)] \tag{32}$$

gives on rewriting:

$$y = \frac{J(u)vg}{1 + J(u)g} + \frac{J(u)y_d g}{1 + J(u)g}. \tag{33}$$

Considering the updating equations of the gain g we have,

$$\frac{\partial E_y}{\partial g} = \frac{\partial E_y}{\partial y}\frac{\partial y}{\partial g}. \tag{34}$$

But

$$\frac{\partial y}{\partial g} = \frac{\partial y}{\partial u}\frac{\partial (zg)}{\partial g}. \tag{35}$$

As

$$\frac{\partial(zg)}{\partial g} = z + g\frac{\partial z}{\partial g} \quad \text{and} \quad \frac{\partial z}{\partial g} = -\frac{\partial y}{\partial g}, \tag{36}$$

$$\frac{\partial y}{\partial g} = \frac{\partial y}{\partial u}\left[z + \left(-\frac{\partial y}{\partial g}\right)g\right]. \tag{37}$$

On rewriting, we obtain:

$$\frac{\partial y}{\partial g} = \frac{(\partial y/\partial u)z}{(1 + g(\partial y/\partial u))} = \frac{J[v + e_y]}{1 + gJ}. \tag{38}$$

Or

$$\frac{\partial E_y}{\partial g} = -\frac{e_y J[v + e_y]}{1 + gJ}. \tag{39}$$

In this case, equation (28) is modified to:

$$g(k) = g(k-1) + ae_y J\frac{[v + e_y]}{1 + gJ}. \tag{40}$$

The important differences between the gain layer scheme presented here and the error transformation schemes [1],[3],[11] are as follows. In the gain layer scheme we use the system error rather than a transformed error. This reduces the rapid variations in the NN weights due to the ill effects present in numerical estimation of the Jacobian. In our case the Jacobian affects only the gain layer weights, and their updating is controlled by a different learning rate. This implies that the controller NN can use a larger learning rate to obtain faster convergence. Our simulation results support this argument. Also, because of the presence of error feedback the system response is faster. The NN time constant plays only a secondary role in the dynamical response of the system. Further, the gain layer is adaptable and the initial value of g need not be estimated precisely, unlike in the feedback gain scheme [3], where the gain is fixed and needs to be estimated precisely for better tracking.

3.3. Extension to MIMO systems

The proposed gain layer schemes (Subsections 3.1 and 3.2) can be readily extended to deal with MIMO dynamics. Figure 6 shows the scheme for a two-input/two-output dynamics. For a general n-input/m-output dynamics, the NN will have m outputs, and the $(n \times m)$-matrix $\mathbf{g}$ will represent a fully connected single layer network with mn weights. The Jacobian of the dynamics will be an $(m \times n)$-matrix with elements

$$J_{ij} = \frac{\partial y_i}{\partial u_i}, \quad i = 1, \ldots, m \text{ and } j = 1, \ldots, n. \tag{41}$$

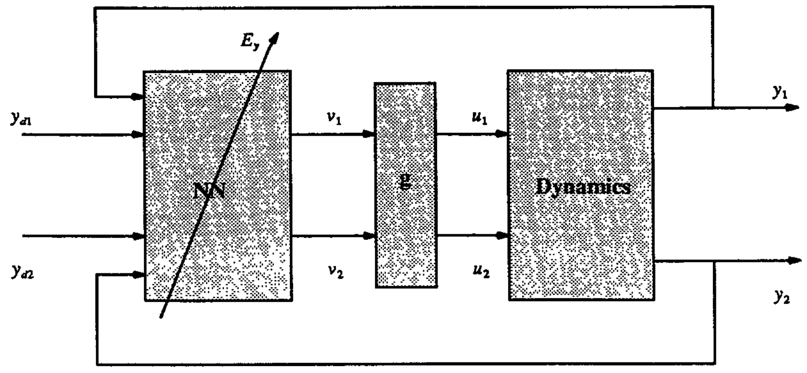

Figure 6. Gain layer scheme for the control of MIMO dynamics.

The NN as well as the gain layer are updated on-line by minimizing the accumulated error E_y over all the outputs of the dynamics. That is,

$$E_y = \frac{1}{2}\sum_{1=1}^{m}(y_{d_i} - y_i)^2 = \frac{1}{2}\|Y_d - Y\|^2. \tag{42a}$$

The weight updating equations for the NN controller are the same as in equations (5) and (6) except that the output layer rule becomes

$$\delta_j^p = (y_{d_j} - y_j)f'(net_j^p). \tag{42b}$$

Proceeding as in Subsections 3.1 and 3.2, after some matrix manipulations for the chain rule of differentiation [13], the vectorized version of the gain updating equations can be obtained for the MIMO scheme. Equation (28) for updating gain layer (Figure 4) is extended to

$$g(k) = g(k-1) + aJ^T[Y_d - Y]V^T \tag{43}$$

where all the data vectors are assumed to be column vectors of appropriate dimensions and T indicates the matrix transpose.

Similarly for the error feedback scheme, equations (33) and (40) are modified to

$$Y = [I_m + Jg]^{-1}Jg[V + Y_d] \tag{44}$$

and

$$g(k) = g(k-1) + aJ^T[I_m + Jg(k-1)]^{-T}[Y_d - Y][V + Y_d - Y]^T \tag{45}$$

where I_m is the $(m \times m)$ identity matrix and $-T$ indicates the inverse followed by transposition. The simulation results for such a scheme are given in the following section.

4. Simulation results

We applied the proposed gain layer scheme to control the nonlinear dynamics of an autonomous underwater vehicle. Assuming that the vehicle is traveling at a constant speed, the input-output relationship of the dynamics can be modeled using a set of fifth order differential equations having nonlinearities up to a degree of four [2]. The objective of the neural network controller is to provide a correct deflection angle δ, to the vehicle "stern plane" actuator to maintain a commanded pitch θ_d [2], [15]. The governing equations of the dynamics are given in the Appendix.

The output states of the vehicle (the linear velocities along the three coordinate axes v_x, v_y, and v_z and angular velocities p, q, and r) are obtained at each instant by integrating and solving the equations representing motion by integration, using a time-step of 0.05 sec. The vehicle pitch (system output) is obtained by integrating the angular velocity q, using the same time step. The typical operating range of the vehicle pitch is $+15$ deg to -15 deg.

4.1. Controlling SISO dynamics

The controller NN weights and the gain layer are updated at every iteration that corresponds to a time step of 0.05 sec. A controller NN of size $2 \times 20 \times 10 \times 1$ is used in all the experiments. The two inputs to the NN are the desired pitch and the actual pitch of the vehicle. The size of the controller NN is selected based on a rough estimate of the complexity of the dynamics. In earlier studies [17],[18], we found that a single hidden layer was insufficient to model the nonlinearity. After experimenting with different numbers of neurons in the hidden layers, we found empirically that a $2 \times 20 \times 10 \times 1$ network provides the best performance.

Simulation studies are conducted for all the five schemes and the performances of the resultant control systems are compared by monitoring the speed of convergence, tracking error, and stability of the system. The controller NN weights are initialized to small random values between $+0.1$ and -0.1 in all the cases. On-line updating of the controller and gain layer weights are continued throughout the experiment. All vehicle states are initialized to zero at the beginning of each simulation study.

Figure 7 shows the results of comparison for a commanded pitch of 8.5 deg. The results shown are the best ones for each method over a number of trials. Scheme (1), which uses transformation of the error by backpropagating it through the dynamics, results in slow tracking of the command signal. The inverse transfer matrix scheme (2) also suffers from poor tracking as the high value of the numerically computed inverse differential gain causes large fluctuations in the transformed error. In both cases, the learning rate used as 0.0001; any attempt to increase it further tends to create unstable responses. The feedback gain scheme (3) and the gain layer schemes (4), (5) were found to provide good tracking performances. The gain layer scheme with no error feedback (5) performs significantly better than the specialized learning scheme of [11] and the inverse transfer matrix scheme [1]. The gain K_k (see Figure 3) for the feedback gain scheme was found from a linearized model of the vehicle dynamics.

To this end, the linearized system functions of pitch (θ) and vertical velocity (v_z) were determined at the operating forward speed of 10.16 ft/sec. They are:

$$\frac{\theta}{\delta_s} = \frac{-(0.07808s^2 + 0.2311s + 0.0048)}{0.5077s^4 + 0.4576s^3 + 0.0780s^2 + 0.0050s + 0.0024} \tag{46}$$

$$\frac{v_z}{\delta_s} = \frac{-(0.4228s^3 + 0.5420s^2 + 0.491s + 0.00799)}{0.5077s^4 + 0.4576s^3 + 0.0780s^2 + 0.0050s + 0.0024}. \tag{47}$$

From equation (46), the inverse of the steady state gain was calculated ($= -0.5$) and K_k was set to this value. The gain layer weight g was also initialized to the same value and updated on-line using a learning rate of 0.001. Equation (47) was used for the MIMO scheme described in the next subsection. The vehicle dynamics described by the nonlinear equations (A1) and (A2) (see Appendix) was used in all the experiments. It can be seen from Figure 7 that the gain layer scheme using the error feedback provided better transient response and smaller tracking error. The convergence of the gain g with respect to time is shown in Figure 8.

In Figure 9 we demonstrate the tracking effectiveness of the gain layer scheme for fast-varying command signals. A command signal of the form

$$y_d(k) = 4\sin\left(\frac{2\pi k}{7000}\right) + 2\sin\left(\frac{2\pi k}{2000}\right) + 0.8\sin\left(\frac{2\pi k}{1000}\right) + 2.8\sin\left(\frac{2\pi k}{3000}\right) \tag{48}$$

was given as the input to the network. The results shown are with a network of size $2 \times 20 \times 10 \times 1$ trained with a learning rate of 0.008. The controller is able to make the system track the command signal closely. The dotted lines correspond to the gain feedback scheme with the error feedback. The gain K_k as well as the initial value of g are kept the same as in the previous simulation. As in the other example, the on-line adaptation in the gain layer scheme drives the system towards better tracking. In all the simulation studies, the controller is trained starting from random initial conditions.

4.2. Controlling MIMO dynamics

We conducted some preliminary simulation studies on the MIMO control by employing the gain layer scheme (without the error feedback) to control a single-input/two-output vehicle dynamics. The controller generates the control signal δ_s (stem plane angle) based on the desired and actual values of the system outputs (pitch θ and vertical velocity v_z.) The scheme is similar to the one in Figure 6 except that there is only one input to the dynamics. The governing equations of the system dynamics are detailed in the Appendix. The NN controller of size $4 \times 20 \times 10 \times 2$ has four input neurons corresponding to the desired and actual outputs of pitch (θ)

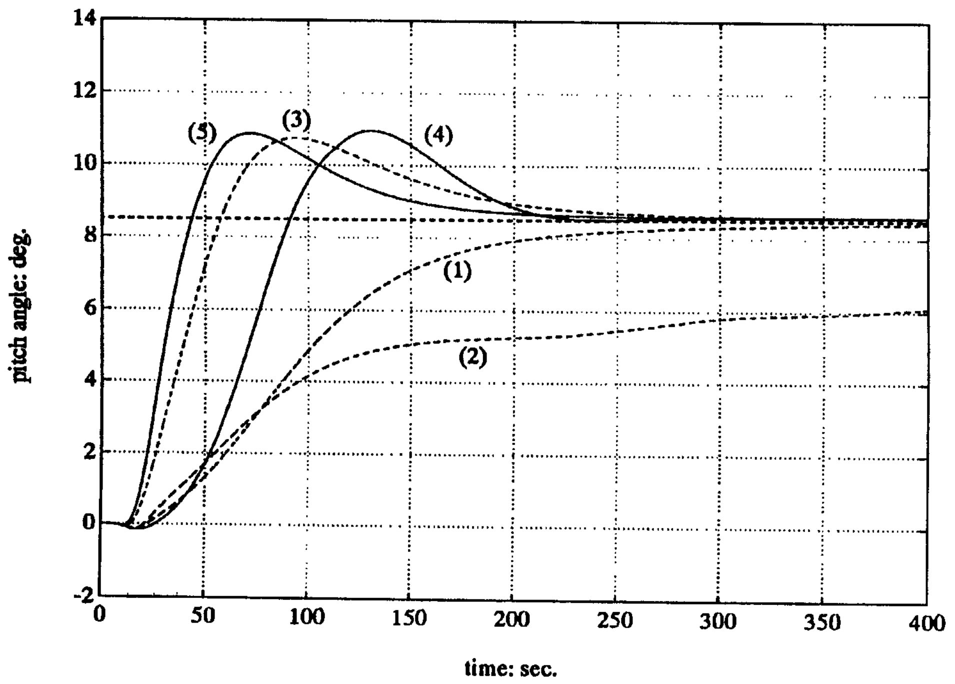

Figure 7. System response for a commanded pitch of 8.5 deg. (1) Specialized learning scheme, (2) Inverse transfer matrix scheme, (3) Feedback gain scheme, (4) Gain layer scheme (without error feedback), (5) Gain layer scheme (with error feedback). A learning rate of 0.0001 was used for schemes (1) and (2). For the feedback gain scheme and the gain layer schemes, the learning rate used for the neural network controller was 0.008. The gain layer was updated using a learning rate of 0.001. All the schemes were compared using a network of size $2 \times 20 \times 10 \times 1$.

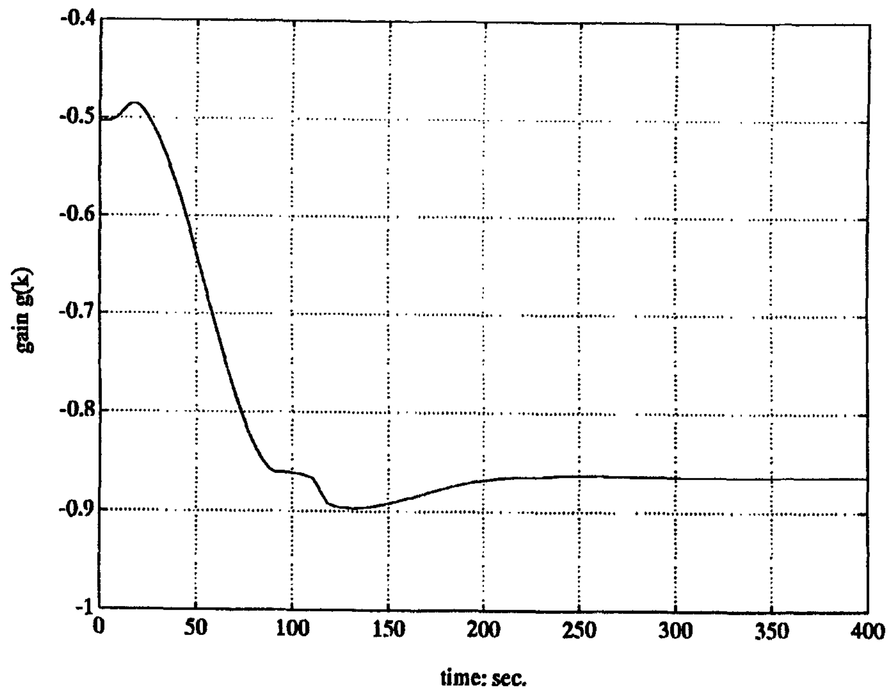

Figure 8. Variation of the gain factor for the response shown in Figure 7. The learning rate used for adapting the gain was 0.001. The learning rate for the controller NN was 0.008.

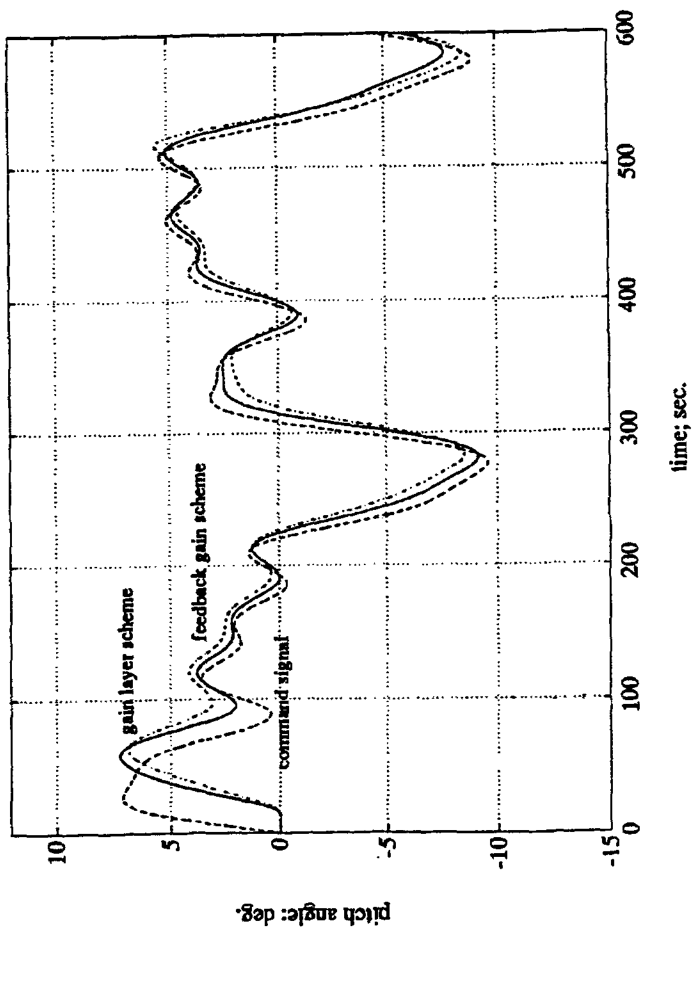

Figure 9. Comparison of the feedback gain scheme and the gain layer scheme for a time-varying command signal. The network learning rate used for both the schemes was 0.005.

and vertical velocity (v_z). The gain layer **g** consists of two weights g_1 and g_2. The NN and the gain layer are updated at each instant by minimizing the combined system error E_y using equations (42) and (49), where

$$E_y = \frac{1}{2}\left[(\theta_d - \theta)^2 + (v_{z_d} - v_s)^2\right]. \tag{49}$$

The results are shown in Figures 10(a and b) for trapezoidally varying command signals θ_d and v_{z_d}. It may be pointed out that these signals cannot be chosen independently because of the coupled nature of the vehicle dynamics. The network learning rate was 0.001 and the learning rate for the gain layer was 0.0001. The weights were updated at each time instant (0.05 sec. apart). The NN weights were initialized to small random values between −0.1 and +0.1, as in the previous simulation studies, and the gain layer weights g_1 and g_2 were initialized tp [−0.5, −0.39] corresponding to the inverse of steady state gain determined from the linearized vehicle dynamics (equations (46) and (47)). From Figure 10, it can be seen that the NN controller was able to provide close tracking performance.

Conclusion

The choice of the neural network controller architecture plays an important role in obtaining the desired dynamic response and tracking performance. In this paper, we have presented an improved neurocontroller architecture for direct control of dynamical systems that uses the backpropagation algorithm effectively. The ill effects of the controlled dynamics on the on-line updating of the NN weights are moderated by including a compensating gain layer. The presented method avoids the transformation needed in other schemes and improves the system response using error feedback. With some examples, we have shown that this scheme performs well and gives better tracking and dynamical responses. Also, the performance of the gain layer scheme was studied on the problem of controlling two output parameters simultaneously.

Acknowledgments

The authors wish to thank Professor Stanley Dunn for his keen interest in this work. They are also grateful to the reviews for their constructive criticisms and suggestions, which helped improve the clarity of this paper.

Appendix

This section provides the governing equations of the autonomous underwater vehicle dynamics employed in the simulation studies [2], [15],[17].

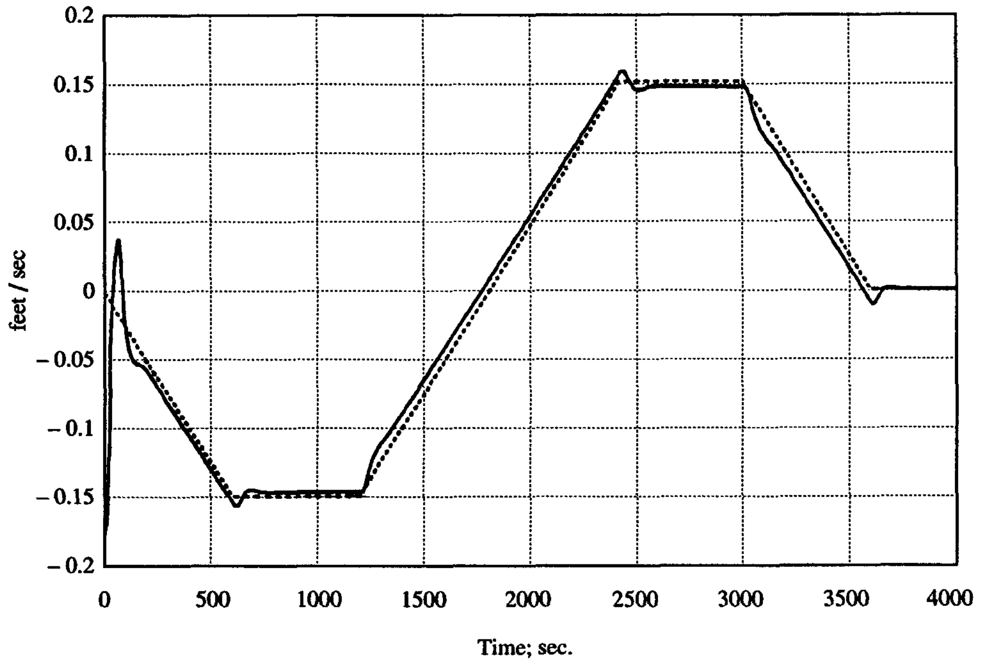

Figure 10a. Simultaneous control of pitch (θ) and vertical velocity (v_z). The NN and gain layer are of sizes $4 \times 20 \times 10 \times 2$ and 2×1, respectively. Learning rate for the NN = 0.001. Learning rate for gain layer = 0.0001. The dotted line represents the command signal; the solid line corresponds to the vehicle response. (a) Pitch angle (deg.).

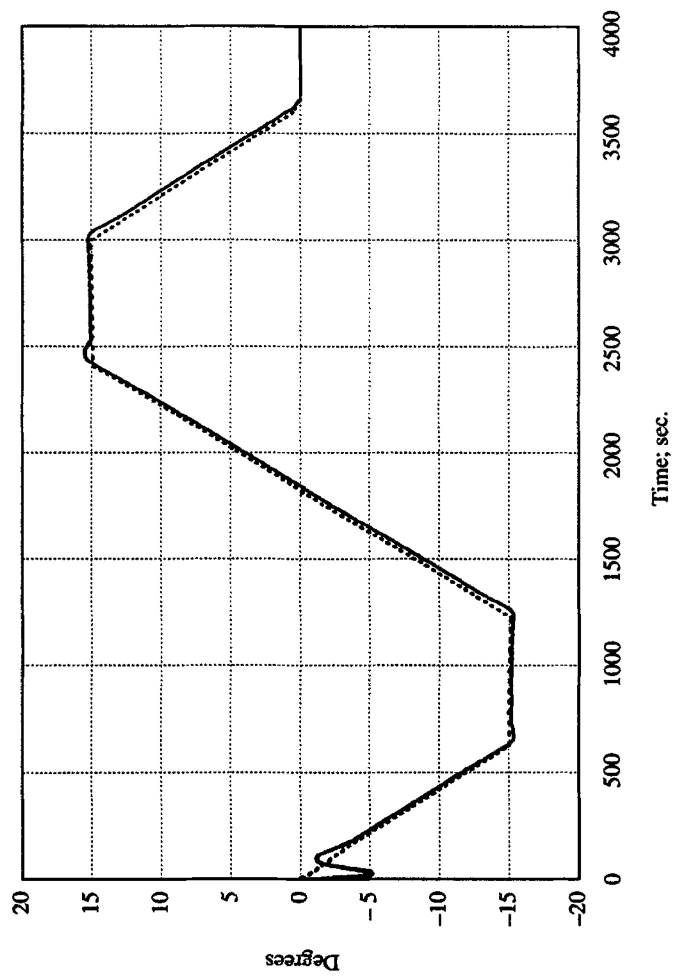

Figure 10b. Vertical velocity (ft/sec.).

A.1. Pitch angle (θ)

The pitching moment of the vehicle derived from the conventional Euler equations of motion is given by

$$\begin{aligned}\sum M_M &= I_y\dot{q} + (I_x - I_y)rp - (\dot{p} + qr)I_{xy} + (p^2 - r^2)I_{xz} + (qp - \dot{r})I_{yz}\\ &\quad + m\big[z_G(\dot{v}_x - v_y r + v_z q) - x_G(\dot{v}_z - v_x q + v_y p)\big] \qquad \text{(A1)}\end{aligned}$$

where v_x, v_y, v_z, $\dot{v}_x$, and $\dot{v}_z$ are the linear velocities and their derivatives along the three coordinate axes x, y, and z, respectively; $p, q, r, \dot{p}, \dot{q}, \dot{r}$ are the angular velocities and their derivatives; and x_G, y, and z_G are the location of the center of gravity with respect to the global coordinate system. The parameters (I_x and I_y) are the moments of inertia with respect to the three coordinate axes; I_{xy}, I_{yz}, and I_{xz} are the moments of inertia with respect to the planes xy, yz and xz, respectively; and m corresponds to the mass of the vehicle.

The summation of all the pitching moments estimated from the vehicle body characteristics and the velocities is given by

$$\begin{aligned}\sum M_M &= 0.5\rho l^5[M'_{\dot{q}}\dot{q} + M'_{rp}rp] + 0.5\rho l^4[M'_{\dot{v}_z}\dot{v}_z + M'_q v_x q]\\ &\quad + 0.5\rho l^3\big[M'_*\dot{v}_x^2 + M'_{v_z}v_x v_z + M'_{v_z|v_z|}v_z\sqrt{v_y^2 + v_z^2}\big]\\ &\quad + 0.5\rho l^3\big[M'_{|v_z|}\dot{v}_x|\dot{v}_z| + M'_{v_z v_z}\sqrt{|v_z(v_y^2 + v_z^2)|}\big]\\ &\quad + 0.5\rho l^3\left[M'_{\delta_s}v_x^2\delta_s + M'_{\delta_s\eta}v_x^2\delta_s\left(\eta - \frac{1}{c}\right)c\right]\\ &\quad + 0.5C_4\int_l xb(x)v_z(x)\sqrt{v_y^2(x) + v_z^2(x)}\,dx\\ &\quad - 0.5\rho l\bar{C}_L\int_{x_2}^{x_1} xv_y(x)\bar{v}_{FW}(t - \tau[x])dx \qquad \text{(A2)}\end{aligned}$$

where ρ is the density of sea water, δ_s is the stern plane deflection (input), and l is the length of the vehicle; other parameters such as M' are various hydrodynamic coefficients determined from the vehicle characteristics. The states of the vehicle (v_x, v_y, v_z, p, q, r) are obtained at each instant by numerically integrating the equations (A1)–(A2) and (A3)–(A4) (provided below). The vehicle pitch θ is obtained by integrating the angular velocity q. A software package has been developed at the Center for Advanced Marine Systems for performing these computations [15].

The corresponding equations for the vertical velocity relationship are given below.

A.2. Vertical velocity (v_z)

The summation of all the vertical velocities acting on the vehicle, according to the Euler equations of motion is as follows:

$$\sum F_z = m\big[v_z' - v_x q + v_y p - z_G(p^2 + q_2) + x_G(rp - q') + y_G(rq + p')\big] \quad \text{(A3)}$$

where the velocities and coordinates of the center of gravity are defined as follows.

The summation of physical velocities is given by

$$\begin{aligned}\sum F_z = {} & 0.5\rho l^4[Z'_{\dot q}\dot q] + 0.5\rho l^3\big[Z'_{\dot v_z}\dot v_z + Z'_q v_x q + Z'_{v_y p} v_y p\big]\\ & + 0.5\rho l^2\big[Z'_{*}\dot v_x^2 + Z'_{v_z} v_x v_z\big]\\ & + 0.5\rho l^2\big[Z'_{|v_z|}\dot v_x|\dot v_z| + Z'_{v_z v_z}\sqrt{|v_z(v_y^2 + v_z^2)|}\big]\\ & + 0.5\rho l^2\left[Z'_{\delta_s} v_x^2\delta_s + Z'_{\delta_s\eta} v_x^2\delta_s\left(\eta - \frac{1}{c}\right)c\right]\\ & - 0.5C_d\int_l b(x)v_z(x)\sqrt{v_y^2(x) + v_z^2(x)}\,dx\\ & - 0.5\rho l\bar C_L\int_{x_2}^{x_1} v_y(x)\bar v_{FW}(t - \tau[x])dx. \end{aligned} \quad \text{(A4)}$$

As before, the states of the vehicle (v_x, v_y, v_z, p, q, r) are obtained at each instant by numerically integrating the equations (A1)–(A2) and (A3)–(A4).

The parameters used in the simulation studies are given below:

$\delta = 1.99$	$I_x = 41.07$	$I_{xy} = -127.05$	$x_G = -0.92397$
$m = 88.79$	$I_y = 2763.3$	$I_{xz} = 0.0$	$y_G = 0.0$
$l = 22$	$I_z = 2764.17$	$I_{yz} = 0.04$	$z_G = 0.1$
$c = 1$	$C_d = 0.741$	$\bar C_L = 1.0$	
$\eta = 1.0$			
$b = 2859.3$			

$$\begin{array}{ll}
M'_{*} = 0.0 & \\
M'_{\dot{q}} = -7.7194 \times 10^{-4} & Z'_{q} = -1.75817 \times 10^{-2} \\
M'_{q} = -8.373 \times 10^{-3} & Z'_{v_z p} = -1.0428 \times 10^{-2} \\
M'_{v_z} = -7.3357 \times 10^{-3} & Z'_{\dot{q}} = -1.31057 \times 10^{-4} \\
M'_{rp} = 7.5103 \times 10^{-4} & Z'_{\dot{v}_z} = -1.07 \times 10^{-2} \\
M'_{\dot{v}_z} = -1.31057 \times 10^{-4} & Z'_{*} = 0.0 \\
M'_{|v_z|} = 0.0 & Z'_{v_z} = -3.6838 \times 10^{-2} \\
M'_{v_z}|v_z| = 0.0 & Z'_{|v_z|} = 0.0 \\
M'_{v_z v_z} = 0.0 & Z'_{v_z v_z} = 0.0 \\
M'_{\delta_s} = -1.02327 \times 10^{-2} & Z'_{\delta_s} = -2.155 \times 10^{-2} \\
M'_{\delta_s \eta} = 0.0 & Z'_{\delta_s \eta} = 0.0
\end{array}$$

References

[1] V. C. Chen and Y. H. Pao, Learning control with neural networks, *Proc. Int. Conf. Robotics and Automation*, 1989, vol. 3, pp. 1448–1453.

[2] J. Feldman, DTNSRDC revised standard submarine equations of motion, David W. Taylor Naval Ship Research and Development Center, *Tech. Report: no. SPD-0393-09*, 1979.

[3] M. Kawato, K. Furukawa, and R. Suzuki, A hierarchical neural network model for control and learning of voluntary movement, *Biological Cybernetics*, vol. 57, pp. 169–185, 1987.

[4] M. S. Lan, Learning tracking controllers for unknown dynamical systems using neural networks, *Proc. IEEE Conf. Systems, Man and Cybernetics*, pp. 29–31, 1990.

[5] F. L. Lewis, K. Liu, and A. Yesildirek, Multilayer neural net robot controller with guaranteed tracking performance, Technical Report, Automation and Robotics Research Institute, University of Texas at Arlington, March 1993.

[6] W. T. Miller, P. J. Werbos, and R. J. Williams (eds.), *Neural Networks for Control*, MIT Press, Cambridge, MA, 1991.

[7] W. T. Miller, R. P. Hewes, F. H. Glanz, and L. G. Kraft, Real-time dynamic control of an industrial manipulator using a neural network based learning controller, *IEEE J. Robotics and Automation*, vol. 6, pp. 1–9, February 1990.

[8] K. S. Narendra and A. M. Annasamy, *Stable Adaptive Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

[9] K. S. Narendra and K. Parthasarathy, Identification control of dynamical systems using neural networks, *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4–27, March 1990.

[10] M. M. Polycarpou and P. A. Ioannaou, Identification and control of nonlinear systems using neural network models: Design and stability analysis, *Technical Report: TR 91-09-01, University of Southern California, Department of Electrical Engineering Systems*, California, 1991.

[11] D. Psaltis, A. Sideris, and A. Yamamura, A multilayered neural network controller, *IEEE Control Systems Magazine*, vol. 4, pp. 17–21, April 1988.

[12] D. E. Rumelhart, J. L. McCelland, and PDP Research Group, *Parallel Distributed Processing, vol. 1: Foundations*, MIT Press, Cambridge, MA, 1986.

[13] A. P. Sage and C. C. White, *Optimum System Control*, Prentice-Hall, Englewood Cliffs, NJ, 1977, pp. 382–383.

[14] R. M. Sanner and J. J. E. Slotine, Direct adaptive control using Gaussian networks, *Nonlinear Systems Laboratory, MIT, Tech. Report: NSL-910303*, March 1991.

[15] A. Shein and J. Kloskie, Development of simulation facilities for the Autonomous Underwater Vehicle research, *Proceedings of the Summer Simulations Conference (SSC)*, pp. 746–748, 1991.

[16] T. Troudet, S. Garg, D. Mattern, and W. Merril, Towards practical control design using neural computation, *NASA Tech. Report TM-103785*, 1991.

[17] K. P. Venugopal, R. Sudhakar, and A. S. Pandya, On-line learning control of autonomous underwater vehicles using feedforward neural networks, *IEEE Journal of Oceanic Engineering*, vol. 17, no. 4, pp. 308–319, October 1992.

[18] K. P. Venugopal, A. S. Pandya, and R. Sudhakar, Adaptive neural network controllers for autonomous underwater vehicles, *Proceedings of ROV'91*, pp. 361–366, 1991.

[19] T. Yabuta and T. Tamada, Neural network controller characteristics with regard to adaptive control, *IEEE Trans. System, Man and Cybernetics*, vol. 22, no. 1, pp. 170–176, January/February 1992.

[20] J. Yuh, A neural network controller for underwater robotic vehicles, *IEEE Journal of Oceanic Engineering*, vol. 15, no. 3, pp. 161–166, July 1990.