# A Survey and Experimental Study of Neural Network AUV Control

**J⌀rgen Lorentz and J. Yuh**
Autonomous Systems Laboratory
Department of Mechanical Engineering
University of Hawaii
Honolulu, HI 96822, USA

## Abstract

This paper presents a survey of neural network controllers for AUVs. A direct adaptive neural net controller incorporating integral action was designed for the heave motion of the ODIN underwater vehicle. The neural net controller is trained on-line by parallel recursive error prediction method and the critic equation. The influence of the various design parameters of the neural net controller was investigated by computer simulation and the controller was also experimentally tested using the ODIN. Results of the computer simulation and experiment are discussed in this paper.

## 1. Introduction

Due to a great increase in underwater activity, particularly in conjunction with offshore petroleum exploration, a large number of underwater vehicles have been designed and built during the last few years. One great problem in the design of autonomous underwater vehicles (AUVs) is the control of these vehicles, which usually demonstrate highly nonlinear and coupled system dynamics [1-3].

Recently, a new and active area of research within control engineering has been that of neural network control [4-10]. Neural networks have a capability of estimating various mathematical functions, including highly nonlinear functions. Furthermore, such networks can in many cases be trained to adapt to changing input-output relationships. Due to this ability, neural networks may have a great potential in control systems for nonlinear and unknown systems, such as AUVs.

Unfortunately, the entrance of neural network technology on the control engineering arena has been tainted by great controversies as to the usefulness of these networks for control purposes. Whereas several authors have portrayed neural networks as a control problem panacea, others are very skeptical. One important reason for this may be the lack of formalized methods and analyses of neural network controller systems, especially regarding the stability of the controllers.

The aim of this study is to investigate the possibilities of using neural networks in the control of AUVs. Previously proposed controllers are briefly reviewed. An on-line neural net controller was investigated through computer simulations and experiments using the ODIN underwater vehicle.

### 1.1 Why Use Neural Networks for Control

There are several properties of neural networks which make them suitable for control purposes:

***Nonlinearity***: The nonlinear nature of neural networks make them particularly well suited for solving complex nonlinear control problems.

***Parallel structure***: The parallel structure of neural networks facilitates the construction of parallel implementation of control systems. This can result in robust and fast processing systems.

***Hardware implementation***: Neural networks can easily be implemented in hardware. A number of integrated circuits (IC) for artificial neural networks (ANN) purposes are available for purchase.

***Multivariable nature***: Their potential ability to correctly map functions with many inputs and outputs make neural networks interesting for the control of multivariable systems.

### 1.2 Off-line vs. On-line Training

A simple, but common way of implementing control systems incorporating neural networks is using offline learning. In this technique, the neural network controller is first trained (which can be compared to the tuning of a conventional controller) and then put to use. The primary advantage of offline training of neural network controllers is the speed of the resulting network. As no weight adjustments take place during the running of the controller, the response of the controller can be very rapid. The major disadvantage of this control method is that the resulting controller is not adaptive. Hence, inaccuracies in the network weights or changes in system parameters are likely to result in poor performance of the controller system.

An alternative to offline training of a neural network controller is continuously updating the neural network weights as the controller is in use. This is what takes place in adaptive, or online trained, neural network controllers. Usually, a measure of the system performance is set up, and

0-7803-3185-0/96$5.00©1996 IEEE

one attempts to adjust the controller weights in a manner which improves this performance, for instance by minimizing some output error. The main challenge in this process is calculating the optimal weight changes from the system input and output as well as the reference trajectory for the system. Furthermore, ensuring the stability of such controllers is a challenging area where substantial work still remains to be done.

## 1.3 Neural Network Control Structures

Several different neural network controller schemes have been suggested and implemented in the past:

***Identification and modeling:*** Due to their ability to map mathematical functions, neural networks have been used for system identification in connection with control problems. It is well known that a neural network with one hidden layer and a sufficient number of neurons can approximate any continuous mathematical function [11-13]. However, the convergence of learning rules to values which represent these desired mappings have yet to be proven for most of the learning algorithms which are in use today. The applications of neural networks to system identification and modeling can be divided into the following approaches: (a) Forward Modeling; (b) Direct Inverse Modeling; and (c) Indirect Inverse Modeling.

***Direct control:*** Instead of using a neural network for estimating the process behavior and using a conventional controller, the complete controller can consist of, or include, a neural network which controls the system by minimizing some error criterion. This can be done in one of several ways: (a) Supervised Control; (b) Direct Inverse Control; (c) Model Reference Control; (d) Critic Control; (e) Internal Model Control; and (f) Predictive Control.

Presentations of various neural network controller structures can be found in [4,5,14].

## 2. Neural Network AUV Control

A major challenge in the design of AUVs is the construction of the control systems. AUVs generally demonstrate highly nonlinear system dynamics. Furthermore, the exact system of such vehicles are often unknown, as the process of obtaining the hydrodynamic coefficients of the vehicles may be tedious and expensive. In spite of the increase in neural network applications in control and robotics during recent years, the work that has been done on neural network controllers for AUVs is quite limited.

## 2.1 Previous Neural Network AUV Controllers

Almost all of the neural network controllers that have been designed for AUVs are direct controllers where the NN forms the main part of the controller. Compared to other AUV control systems, the NN control schemes are in general quite crude, many being SISO or offline trained controllers. Offline trained, nonadaptive AUV controllers have been designed by Porto and Fogel, and Sanner and Akin. Online controllers have been proposed by Yuh et. al., Venugopal, Sanner, and Ura et. al.

*Porto and Fogel's AUV Controller*

In 1992, Porto and Fogel, of ORINCON Corp, used several primitive, offline trained neural network structures for predicting the motion of an AUV [15]. They regarded AUV motion in the X-Y plane, using position and velocity in the X- and Y-directions as the four inputs to the neural networks. Their neural networks had two outputs, the desired change in the velocities in the X- and Y-directions. Hence, the neural networks trained were not actually controllers, as claimed by Porto and Fogel, but state estimators.

Their training data consisted of scheduled way points on a multidimensional grid, giving desired neuron outputs for specified inputs. Both straight-line and circular AUV motion way points were presented to the neural networks. The training algorithms used were ordinary back-propagation (BP) [16] and evolutionary programming. In evolutionary programming, a parent weight vector is changed randomly, giving several children weight vectors. Of these vectors, the ones with the best performance, i.e. the lowest square error, are chosen and the evolution is continued in the same manner until sufficiently good network performance is attained. Porto and Fogel performed no actual AUV experiments, but they were able to teach their neural networks their desired trajectories. The paper written by Porto and Fogel does not state how they intended to use their state estimator neural networks for actual vehicle control purposes.

*Sanner and Akin's Pitch Attitude Regulator*

Sanner and Akin, of the MIT Space Systems Laboratory, implemented a pitch controller for an underwater telerobot using a three-layer neural network in 1990 [17]. Their network consisted of two linear input neurons (with inputs pitch angle and velocity), three sigmoid hidden neurons, and one linear output neuron, outputting the actuator input.

The NN was trained using a simulated model of the plant using BP. The output error of the NN controller was estimated using the critic equation:

$$e(t) = M_\theta(\theta_d - \theta) + M_u u^*(t) \qquad (1)$$

where $\theta_d$ is the desired pitch angle; $\theta$ is the actual pitch angle; $M_\theta$ is a weighting matrix; $M_u$ is a multiple of the identity matrix; $u^*(t) = -\left(\frac{u_i(t)}{u_{ult,i}}\right)^n$ is a normalized control vector; $u_{ult,i}$ is a normalizing factor, typically the actuator saturation value; and n is an odd integer.

The controller was trained offline. After the training of the NN controller network had been completed, the trained controller was connected to the telerobot and tested experimentally, with quite good results. Sanner and Akin observed that the NN, after training, implemented a switching layer controller.

***Yuh's Gradient Descent Controllers***

Yuh, of the Autonomous Systems Laboratory, University of Hawaii at Manoa, has proposed several adaptive critic NN AUV controllers using the backpropagation (BP) and parallel recursive error prediction method (PRPEM) learning algorithms [18-21].

The controllers suggested by Yuh are adaptive, online trained critic controllers. The inputs to the controllers are the position error vector, and the velocity error vector, as well as a bias. The neural network output is the control output (force vector).

Unlike most of the other earlier proposed NN controllers, Yuh's NN controllers are not trained to mimic a predetermined desired controller action. Instead, he proposes training the controllers using one of two gradient descent learning algorithms, BP or PRPEM, and using an estimate of the neural network output error based on the tracking error of the vehicle through a critic equation. The estimate he suggests is of the form:

$$\Delta U(t) = U^*(t) - U(t) = C\left[e_v(t) + \lambda_1 e_x(t)\right] \tag{2}$$

where ΔU(t) is an estimate of the NN output error; C is given by the equation $C = \frac{M_{est}}{T}$; T is the sampling time for the controller; and $\lambda_1$ is a cost weighting parameter.

(1) Yuh's BP Controller

In 1990, Yuh designed an adaptive BP controller without momentum learning using this estimate of the output error [18]. This resulted in the following weight update law:

$$W_m(t+1) = W_m(t) + \rho\delta_m q_{m-1}{}^T \tag{3}$$

Here, $W_m(t)$ is the weight matrix between the (m-1)th and m-th layer and $q_m$ is the vector of neuron outputs from the th layer. The vector $\delta_m$ is given by:

$$\delta_m = \begin{cases} f_m'(p_m)\Delta U & \text{for the output layer} \\ f_m'(p_m)W_{m+1}^T\delta_{m+1} & \text{for the other layers} \end{cases} \tag{4}$$

Here, $f'(x)$ is the derivative of the neural network activation function. For this controller, Yuh used the hyperbolic tangent activation function.

(2) Yuh's PRPEM Controller

In 1993 and 1994, Yuh suggested a modification of the controller by exchanging the BP training algorithm with the PRPEM algorithm [22]. This gives the weight matrix update law:

$$W_m(t) = W_m(t-1) + \rho F_i(t)\Psi(t)e(t) \tag{5}$$

where $F_i$ is the approximate inverse of the Hessian matrix and it is computed by using a constant trace method. (see eqs. 16 and 17 in sec. 3) If the inverse hessian matrix $F_i$ is replaced by the identity matrix in eq. 5, the simpler BP controller algorithm in eq. 3 results.

***Venugopal's Gradient Descent Controller***

Venugopal et al. [23] suggest an enhancement of Yuh's BP controller. Rather than estimating the neural network output errors by equation 4.2, Venugopal suggests introducing a variable gain factor 'g' between the neural network controller and the system dynamics. The gain factor 'g' is adjusted and should, ideally, equal the inverse of the system's Jacobian. The algorithm which Venugopal suggests for updating the gain factor is:

$$g(t) = g(t-1) - \alpha\frac{\partial e(t)}{\partial g(t)} = g(t-1) + \alpha\left[y_d(t) - y(t)\right]\frac{\partial\delta y(t)}{\partial\delta u(t)}\upsilon(t) \tag{6}$$

where g(t) is the gain factor; α is an updating rate constant; $y_d(t)$ is the desired system output; y(t) is the actual system output; u(t) is the system input; and v is the NN output.

The neural network is then updated by backpropagating the system output error through the neural network. For MIMO systems, Venugopal et. al. claim that the gain factor 'g' becomes a diagonal matrix. This, however, implies disregarding the interconnections of the systems various inputs and outputs in updating the NN weights, which may be an oversimplification if the system dynamics contain significant interactions between the various directions of motion.

Venugopal et. al. further propose extending the AUV controller to multiple-degrees-of-freedom by using a separate neural network for each degree of freedom. The paper written by Venugopal et. al. includes promising simulation results for a MIMO controller.

*Ura's NN AUV Control Systems*
Ura and his colleagues at the URA Lab, University of Tokyo, have worked on several indirect inverse modeling neural network AUV controllers which they have tested on their Twin-Burger underwater vehicle.

(1) The SONCS Self-Organizing Control System
In 1990, Ura suggested SONCS (Self-Organizing Neural-Net Control System), which is an indirect inverse modeling controller consisting of a controller network and a forward modeling network [24].

Ura trained the forward modeling network offline using experimental data from the vehicle. The controller network was then trained online using backpropagation. The output errors of the controller network were estimated by backpropagating the system output error through the forward modeling network. Hence, the forward model provides an estimate for the NN controller output error.

(2) SONCS With Imaginary Training
In 1994, Ura proposed an improvement to the SONCS system which he labeled imaginary training. The new control system consists of two parts, a real world part and an imaginary world part [25, 26]. The real world part consists of the actual controller network $C_R$ and the AUV system dynamics h(s), whereas the imaginary world part consists of an imaginary controller network $C_I$ and the NN forward model.

The imaginary controller network is trained with a high weight updating frequency using backpropagation through the feedforward and $C_I$ networks in simulation. At certain intervals, the imaginary controller network is downloaded to the actual real-world controller. Hence, the weights of the real-world controller network are updated. If the dynamics of the vehicle are significantly altered, the weights of the feedforward network has to be updated through further training.

*Seube's Viability Theory Controller*
A different approach has been used by Seube, who has simulated an adaptive, one-layer NN controller derived from viability theory [27-30]. An introduction to viability theory is given by Aubin in [31]. The main contribution of Seube's work is replacing the optimization criterion of the gradient descent methods by a viability learning constraint. Seube's paper does not state clearly how the selection of the viability constraint and control law should be undertaken. Seube has, nonetheless, performed several computer simulations with this controller and obtained quite impressive results for the surge/sway/yaw control of an AUV. For this control task, the viability controller outperformed a BP controller. Unfortunately, the description of the simulations which were completed is not sufficiently detailed for repetition or further pursuit of these simulations.

A comparison of the previously described neural network controllers for AUVs is shown in Table 1.

Table 1: Comparison of NN AUV Controllers

| Author | System | Training | NN output error estimation | Exp. |
|---|---|---|---|---|
| Porto | MIMO estimator | off-line BP | N/A | no |
| Sanner | SISO | off-line, BP | Critic eq. | yes |
| Yuh | SISO MIMO | on-line recur. BP, PRPEM | Critic eq. | no |
| Venugopal | SISO MIMO | on-line recur. BP | variable gain factor | no |
| Ura | MIMO | SONCS BP | BP thru forward NN | yes |
| Seube | MIMO | on-line Viability | N/A | no |

## 2.2 Suggested Neural Network AUV Controllers
The majority of these proposed neural network controllers for AUVs are of the gradient descent type. These controllers seem to be the ones that are best understood, and with the possible exception of Seube's viability controller, the ones which have given the most promising results. Therefore, a more thorough investigation of such controllers has been conducted by computer simulation and experiment. The SISO controllers which have been designed are closely based on the earlier work performed by Yuh and Venugopal et. al.

Important properties of these controllers and networks include:

- Direct controllers: The neural networks perform the actual control of the vehicle.
- Online learning: All the controllers are designed to be trained online. The motivation for this is that a minimal amount of a priori system knowledge is required and that the controllers should have an ability to adapt to changes in the system dynamics.

- Feedforward networks: Only feedforward networks are used. Feedback is introduced in the controller design by using system outputs as the network inputs.

The neural network having three layers (an input layer, one hidden layer, and an output layer) was considered. The neural network inputs were:

- The velocity error $e_\omega(t) = \omega_d(t) - \omega(t)$
- The position error $e_z(t) = z_d(t) - z(t)$
- The integral of the position error $e_i(t) = \int_0^t e_z(\tau)d\tau$
- A bias input, d

The introduction of the integral error as a NN controller input was inspired by regular PID control system design. The 3-layer network used for experiment can be described by the following equation:

$$u = q_3 = f_3(p_3) = f_3(W_3 f_2(W_2 e + W_{2d} d) + W_{3d} d) \qquad (7)$$

where $e = q_1$ is the error input vector $\left[\omega_d(t) - \omega(t), z_d(t) - z(t), \int_0^t (z_d(\tau) - z(\tau))d\tau\right]^T$; $p_2$ is the activation vector of the neurons in layer 2; $p_3$ is the activation of the neuron in layer 3; u is the network control output; $f_2$ is the vector activation function of layer 2; $f_3$ is the activation function of layer 3; $W_2$ is the $N_2 \times N_1$ dimensional weight matrix between layers 1 and 2; $W_3$ is the $N_3 \times 1$ dim. weight matrix between layers 2 and 3.

The activation functions $f_2$ and $f_3$ are given by

$$f_2(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \qquad (8)$$

$$f_3(x) = \frac{F_{MAX} + F_{MIN}}{2} + \frac{F_{MAX} - F_{MIN}}{2} \cdot \frac{1 - e^{-x}}{1 + e^{-x}} \qquad (9)$$

The vehicle model can be represented by the following equation:

$$(m + Z_{\dot\omega})\dot\omega + Z_{\omega|\omega|}|\omega|\omega + (\rho V - m)g = F_z \qquad (10)$$

where mass, m=122.49 kg; added mass in heave, $Z_{\dot\omega}$=21.87 kg; hydrodynamic damping, $Z_{\omega|\omega|}$=48 kg/m; water density, $\rho$=1025 kg/m$^3$; vehicle volume, V=0.1194 m$^3$; and gravitational acceleration, g=9.81 m/s$^2$.

The actuators for the heave control of ODIN are four vertical thrusters. The thrusters operate in the interval ±3000rpm. The force from an individual thruster is assumed given by the equation:

$$F_T = C_T \omega_m \qquad (11)$$

where $C_T$ is a motor constant and $\omega_m$ is the motors angular velocity in rad/s. The thruster has an asymmetric characteristic, and the motor constant $C_T$ is given by [32]:

$$C_T = \begin{cases} 6.835 \cdot 10^{-5} \ \ N/(rad/s)^2 & \text{if } \omega_m \geq 0 \\ -2.339 \cdot 10^{-5} \ \ N/(rad/s)^2 & \text{if } \omega_m < 0 \end{cases} \qquad (12)$$

## 3. Computer Simulations

In the simulations of the presented NN control system, performed with the heave motion model of ODIN, the following design issues of the neural network controller for the vehicle were investigated:

***Choice of Learning Algorithm***

During the computer simulations of the SISO NN control system for the heave motion of ODIN, two different learning algorithms were tested: the backpropagation (BP) algorithm; and the parallel recursive prediction error method (PRPEM) summarized below:

$$\text{BP:} \qquad \Theta(t) = \Theta(t-1) + \Delta(t) \qquad (13)$$

$$\text{PRPEM:} \qquad \Theta(t) = \Theta(t-1) + F(t)\Delta(t) \qquad (14)$$

$$\Delta(t) = \gamma\Delta(t-1) + (1-\gamma)\rho\Psi(t)e(t) \qquad (15)$$

where $\Theta$ is a vector including all weight matrix elements and the matrix $\Psi$ is the vector derivative of the output error with respect to the weight vector;

$$\begin{aligned}\overline{F_i}(t+1) &= F_i(t) \\ &- F_i(t)\Psi_i(t+1)\left(\lambda I + \Psi_i^T(t+1)F_i(t)\Psi_i(t+1)\right)^{-1}\Psi_i^T(t+1)F_i(t)\end{aligned} \qquad (16)$$

$$F_i(t+1) = \frac{\alpha\overline{F_i}(t+1)}{\mathrm{trace}\left(\overline{F_i}(t+1)\right)} \qquad (17)$$

Both the PRPEM and BP algorithms were able to give acceptable controller performance for this system. It was found that the PRPEM algorithm yielded the best result for the tested reference trajectory when integral action was used, whereas the BP algorithm could not be made stable when the integral error was used as one of the controller inputs. When th best BP and PRPEM controllers were compared, it was found that the BP controller demonstrated better tracking (smaller MSE) for the transient part of the trajectory. For stationkeeping, the PRPEM controller was superior, probably because of its integral action. It was also found that the BP controllers were generally less stable than the PRPEM controllers, which may be one reason why the integral action could not be incorporated into the BP controllers without causing instability.

*Use of Integral Action*

Earlier simulations [33] have shown that a serious direct gradient descent NN controllers of the kind investigated here can be steady-state errors in the tracking performance of the controlled systems. In order to solve this problem, several NN controller schemes which incorporate integral action were tested. Computer simulation tested two controllers: one uses the integral error as a NN input and a term in the neural network output error estimate; the other does not. Except for this, the controllers are identical. The effects of incorporating integral action into the NN controller can hence be summarized as:

- Significant reduction of steady-state errors.
- Better tracking of the desired position profile.
- Slightly poorer tracking of the desired velocity profile. (It is for this testing case, because of the initial oscillations of the IA controller depth about the desired depth value.)
- Slight increase in the computational requirements of the controller network.

*Influence of Controller Parameters*

The performed simulations and experiments demonstrated that it is possible to control the heave motion of the ODIN vehicle using a SISO NN controller utilizing either the BP or PRPEM learning algorithms. It was also found that the performance of the tested controllers was highly dependent on the choice of controller parameters:

- Learning constants $\rho_2$ and $\rho_3$
- Network bias value d
- Momentum constant $\gamma$
- Error function weighting parameters $\lambda_1$ and $\lambda_2$

Unfortunately, the selection and tuning of these parameters has previously been a largely ad hoc trial and error process, In order to increase the utility of this kind of controllers, it is desirable to establish a set of rules for selecting the various controller parameters and thus reducing or eliminating on-site controller tuning requirements.

***Learning Constants:*** The effects of changing the two learning constants $\rho_2$ and $\rho_3$ were investigated through computer simulations whose results are summarized below:

- Improved controller performance may be obtained by using different learning constants for the different network layers. In this case, the best performance resulted when the learning constant $\rho_2$ was approximately 10 times larger than $\rho_3$.
- In order to ensure convergence toward zero tracking error, very small learning rates should be avoided. The reason for this may be the existence of local minimum of the error function.
- Higher values for the learning constants should be used for a transient reference trajectory than for stationkeeping.

***Network Bias Value:*** In neural networks, it is customary to add a bias value to the activation of each neuron. The reason for this is that a neural network with bias inputs generally has a potential for approximating a wider range of different mathematical functions than one which does not use a bias. This can be understood by considering the n inputs to a neuron in a network as spanning an n-dimensional space. The neuron will then define a hyperplane through this space, with a zero output value on the hyperplane, a positive output value on one side of the hyperplane, and a negative output value on the side. If a bias input is not used, this hyperplane will necessarily pass through the origin of the n-dimensional space. If a bias value is used, hyperplane can be shifted away from the origin [34].

If the bias inputs d are all set equal to zero, the neural network output u will be zero if $e(i) = 0, \quad \forall i \in 1 \ldots N_k$ where $N_k$ is the number of neural network inputs.

For the controller network in question, this means that the commanded force from the controller will be zero if the position, velocity, and integral position errors are all zero, which seems reasonable. However, this may be a problem if a nonzero force is required to keep the controller moving along its desired trajectory, for example, if gravitational forces are present. In that case, it could be desirable to lift the control hyperplane away form the origin of the 3-dim. space spanned by the tracking errors $e_v$, $e_x$, and $e_i$.

The influence of the bias input value *d* on the performance of the neural network controller was analyzed by simulating controllers using different biases. No clear relationship between the value of the bias input and the output errors could be found, but the best results were generally obtained for very small values of the bias input (0 or 0.005). A trend which was found is that as the bias value was increased, the setpoint position tracking performance improved whereas the setpoint velocity tracking grew poorer.

***Momentum Constant:*** In the recursive variants of the BP algorithm the new search direction at time t, $\Delta(t)$ (eq. 15) is a weighted vector sum of the previous search direction (t - 1) and the steepest descent direction.
In the PRPEM algorithm this search direction is multiplied by the approximate inverse Lessian matrix F to obtain a Gauss-Newton search direction.

The absolute weighting between the steepest descent direction and the previous search direction depends on the momentum constant $\gamma$. The effect of this weighting constant was investigated through simulations using different values for $\gamma$. These results can be summarized as follows:

- The use of momentum learning improves the performance of the gradient descent NN controllers.
- Values for the momentum constants in the interval $\gamma \in < 0.25, 0.9 >$ yielded the best results.

*Error Function Weighting Parameters:* During the neural network controller learning, the output error of the controller network is estimated as:

$$e(t) = U^*(t) - U(t)$$
$$= C\left[\omega_d(t) - \omega(t) + \lambda_1(z_d(t) - z(t)) + \lambda_2\int_0^t (z_d(\tau) - z(\tau))d\tau\right] \quad (18)$$

Obviously, different choices for the coefficients $\lambda_1$ and $\lambda_2$, which determine the relative weighting of the position and integral errors, respectively, to the velocity error, will result in varying controller performance. The way the choice of these parameters affects the NN controller was investigated through computer simulations. The results can be summarized as:

- The best transient and stationkeeping response, and the quickest adaptation of the controller, is obtained with values of $\lambda_1$ between 0.25 and 2.0, that is when the weighting of the position error is of the same order of magnitude as the weighting of the velocity error.
- Large values of $\lambda_1$ lead to instability.
- Large values of $\lambda_2$ result in a poorer transient response without a distinguishable corresponding improvement in the stationkeeping performance. It therefore appears as if a small value can be used $\lambda$.

## 4. Experiments

The computer simulations which were performed with the various SISO NN controllers for the heave motion of ODIN gave quite promising results indicating that neural network controllers of this kind may be well suited for control tasks of this kind. In order to ensure that the neural network controllers designed for the heave motion of ODIN really were able to control the vehicle, some of the implemented controllers were tested experimentally after completion of the computer simulations.

The practical heave control experiments performed with ODIN for testing the neural network controller schemes took place in the diving pool at the University of Hawaii at Manoa. The pool is 17 feet (5.18 meters) deep. ODIN's depth sensor is located in the upper part of the vehicle, and the maximum depth for the vehicle in the pool is thus 15 feet (4.57 meters). Figure 1 shows one of the results obtained in these experiments with PRPEM method for the heave motion of the ODIN. Other results are not included due to limited space.

Results of the experiment agree with those of the computer simulation. The experimental testing of the NN heave controllers for ODIN can be summarized:

- The gradient descent NN controllers were able to control the vehicle.
- When no integral action was incorporated into the controller design, a considerable steady-state position error resulted.
- The introduction of the integral error as a neural network input, or the introduction of an integral term in parallel with the NN controller solved the steady-state error problem.

## Acknowledgments

Funding for this research was provided in part by the NSF PYI Award Grant No. BES91-57896, in part by ONR through Florida Atlantic University, in part by ERC-ACI, Seoul National University, and in part by the NOAA Office of Sea Grant, Dept. of Commerce (R/ES-1). This is Sea Grant Publication UNIHI-SEAGRANT-CP-96-07.

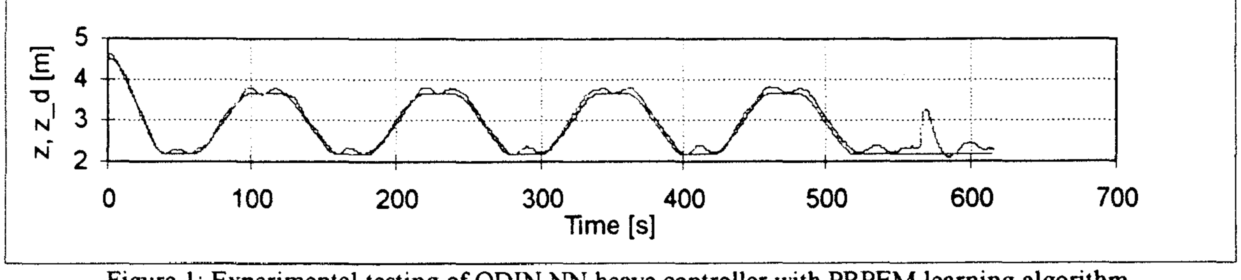

Figure 1: Experimental testing of ODIN NN heave controller with PRPEM learning algorithm.
( ...... actual depth, ____desired depth)

## References

[1] Yuh, J., Modeling and Control of Underwater Robotic Vehicles, IEEE Trans. Sys., Man and Cyber., Vol. 20, No. 6., 1990

[2] H. Mahesh, J. Yuh and R. Lakshmi, A Coordinated Control of an Underwater Vehicle and Robotic Manipulator," J. of Robotic Systems, 8(3), pp. 339-370, 1991.

[3] T.I. Fossen, Nonlinear Modeling and Control of Underwater Vehicles." Dr. Ing Thesis, Norwegian Institute of Technology,1991.

[4] Bekey, G. A. & Gldberg, K. Y. (ed) Neural Networks in Robotics. Boston: Kluwer, 1993.

[5] K.J. Hunt, D. Sbarbaro, R. Zbikowski, and P.J. Gawthrop, Neural Networks for Control Systems-A Survey," Automatica, V.28, No.6, pp.1083-1112, 1992.

[6] W.T. Miller, R.S. Sutton, and P.J. Werbos, Neural Networks for Control, MIT Press, 1990.

[7] IEEE Control System Magazine, April,1988,89,90,92.

[8] Hunt, K. J., Irwin, G. R., & Warwick, K. (eds) Neural Network Engineering in Dynamic Control Systems. London: Springer-Verlag London Ltd, 1995.

[9] Narendra, K. S. & Mukhopadhyay, S. Intelligent Control Using Neural Networks. IEEE Control Systems Magazine, April 1992.

[10] Sanner, R. M. & Slotline, J. J. Gaussian Networks for Direct Adaptive Control. IEEE Transactions on Neural Networks, Vol. 3, No. 6, November 1992.

[11] Chen, S., Billings, S. A., & Grant, P. M. Non-linear system identification using neural networks. International Journal of Control, Vol. 51, No. 6, 1990.

[12] Cybenko, G. Approximations by Superpositions of a Sigmoidal Function, Mathematics of Control, Signals, and Systems, Vol. 2, pp. 303-314, 1989.

[13] Funahashi, K. On the Approximate Realization of Continuous Mappings by Neural Networks, Neural Networks, Vol. 2, pp. 183-192, 1989.

[14] Yuh, J. On Neural Net Controllers for Robotic Manipulators, in Jamshidi, M., Lumia, R., Mullins, J., & Shahinpoor, M. (eds). Robotics and Manufacturing, Vol. 4. New York: ASME Press, pp. 757-762.

[15] Porto, V. W. & Fogel, D. B. Neural Networks for AUV Guidance Control. Initial Efforts Encouraging in Design, Testing of a Two-Dimensional Controller for Accurate Navigation. Sea Technology, February 1992.

[16] D.E. Rumelhart, et al., Parallel Distributed Processing, MIT Press, 1986.

[17] Sanner, R. M. & Akin, D. L. Neuromorphic Pitch Attitude Regulation of an Underwater Telerobot. IEEE Control Systems Magazine, April 1990.

[18] J. Yuh, A Neural Net Controller for Underwater Robotic Vehicles," IEEE J. Oceanic Engineering, Vol. 15, No. 3, pp. 161-166, 1990.

[19] Yuh, J, Learning Control for Underwater Robotic Vehicles, IEEE Control System Magazine, Vol. 14, No. 2, pp. 39-46, 1994

[20] J. Yuh and K.V. Gonugunta, Learning Control of Underwater Robotic Vehicles," Proc. IEEE Int'l Conf. on Robotics and Automation, pp. 106-111, 1993.

[21] J. Yuh. and R. Lakshmi, An Intelligent Control System for Remotely Operated Vehicles," J. of IEEE Oceanic Engineering, V.18, No.1, pp.55-62, Jan. 1993.

[22] S. Chen, C. Cowan, S.A. Billings, and P.M. Grant, Parallel Recursive Prediction Error Algorithm for Training Layered Neural Networks," International Journal of Control, V. 51, No. 6, pp.1215-1228, 1990.

[23] Venugopal, K. P., Sudhakar, R., & Pandya, A. S. On-Line Learning Control of Autonomous Underwater Vehicles Using Feedforward Neural Networks. IEEE Journal of Oceanic Engineering, V. 17, October 1992.

[24] Ura, T., Fujii, T., Nose, Y., & Kuroda, Y. Self-Organizing Control System for Underwater Vehicles. Conference Proceedings. OCEANS '90, Washington, D.C., September 24-26, 1990, pp. 76-81.

[25] Ishii, K., Fujii, T., & Ura, T. An On-Line Adaptation Method in a Neural Network Based Control System of AUV's. IEEE Journal of Oceanic Engineering, Vol. 20, No. 3, July 1995.

[26] Ura, T. & Suto, T. Unsupervised Learning System For Vehicle Guidance Constructed With Neural Network. Proceedings of the 7th International Symposium on Unmanned Untethered Submersible Technology, September 23-25, 1991, pp. 203-212.

[27] Seube, N. Neural Network Learning Rules for Control: Application to AUV Tracking Control. Proceedings of the IEEE Conference on Neural Networks for Ocean Engineering, Washington, D. C., August 15-17, 1991.

[28] Seube, N. A Neural Network Approach For Autonomous Underwater Vehicle Control Based on Viability Theory. Proceedings of the 7th International Symposium on Unmanned Untethered Submersible Technology, September 23-25, 1991, pp. 191-202.

[29] Seube, N. Neural Networks Learning Rules for Control: Uniform Dynamic Backpropagation, Heavy Adaptive Learning Rule. in ref. [4].

[30] Seube, N. Apprentissage de lois de controle regulant des contraintes sur l'etat par resaux de neurones. Comptes Rendus de l'Academie des Sciences de Paris, Serie I, 1991, pp. 445-450.

[31] Aubin, J. P., Viability Theory. Boston: Birkhauser, 1991.

[32] Choi, S. K., Yuh, J., & Takashige, G. Y. Development of the Omni-Directional Intelligent Navigator. IEEE Robotics & Automation Magazine, March 1995, pp. 44-53.

[33] Lorentz, J. Design of a Neural Network AUV Controller. Project work. Norwegian Institute of Technology, May 1995.

[34] Prechelt, L. (maintainer) Frequently Asked Questions (FAQ) on Neural Networks. FAQ for the Usenet newsgroup comp.ai.neural-nets, URL: http://wwwipd.ira.uka.de/~prechelt/FAQ/neural-net-faq.html, April 28, 1995.