An On-Line Trained Adaptive Neural Controller

Yao Zhang, Pratyush Sen, and Grant E. Hearn

This article presents a neural network approach for on-line industrial tracking control applications. In comparison to several existing neural control schemes, the proposed direct neural controller is characterized by the simplicity of its structure and its practical applicability for real-time implementation. In order to enhance the adaptive ability of the neural controller, a set of fuzzy rules is set up for selecting interim training targets. With minor qualitative knowledge about the plant, the scheme is designed for controlling the nonlinear behavior of the plant under conditions of disturbances and noise. Simulations of a ship course-keeping control under random wind forces and measurement noise have been investigated and comparison of performance has been made with a conventional PID controller. Results presented clearly demonstrate the feasibility and adaptive property of the proposed scheme.

Introduction

In recent years, there has been an expansive growth in interest in neural networks over a spectrum of research domains. Within control engineering, neural networks are attractive because they hold the promise of solving problems that have so far been difficult to handle with classical analytical methods. Among the various available neural control schemes, the backpropagation-based neural controllers are probably the most widely applied to tracking control problems. Before discussing our approach in detail, a short review of other existing backpropagation control schemes will be given next.

The architecture proposed by Psaltis et al. [1], called general learning, populates the input space of the plant with training samples so that the network can interpolate for intermediate points. Another control scheme, the specialized learning, learns by directly evaluating the accuracy of the network, that is, the error between the actual and desired output of the plant is used to update the connective weights in the network. In this sense, the controller learns continuously, and hence it can control plants with time-varying characteristics. There are two strategies to facilitate the specialized learning, one being direct control and the other indirect control. In the former, the plant can be viewed as an additional but not modifiable layer of the neural network. The latter, which has been used in many applications [2-4], is a two-step process including identification of dynamics and plant control.

An early version of this paper was presented at the International Conference on Maneuvering and Control of Marine Craft, Southampton, U.K., Sept. 7-9, 1994. The authors are with the Department of Marine Technology, University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, U.K.

There are, however, many unsolved theoretical and practical problems when designing a neural controller to interface with the real world. Neural networks are often described as facilitators for designing adaptive controllers, but it seems that most neural algorithms used for control tasks do not have the property of true adaptive control. The main reason for this is that an off-line network training phase is often required for estimating plant models in order to accomplish the control task.

In general learning, for example, the network needs to be trained off-line to learn the inverse dynamics of the plant. In this case, either the network has to be trained to learn the responses of the plant over a larger operational range than is actually necessary, or the off-line training, no matter how long it is, cannot guarantee the inclusion of all possible situations that could occur in the future. In addition, network training based exclusively on the inverse control error approach generally leads to a bad result, and in practice it has been observed that the output of the neural controller tends to stick at some constant value, resulting in zero training error, but obviously poor control performance [2].

In the indirect control strategy a sub-network (called "emulator" or "mapping neural network") is required to be trained before the control phase, and the quality of the trained emulator is crucial to the controlling performance. It has been observed in our work that the output of the neural controller usually covers a wide range of values at the beginning of the controller's training. Consequently, if some of these values are outside the input range that was used during the emulator's training, the backpropagation through the emulator fails, causing poor or even unstable control performance. It is therefore very important that the data set for training the emulator must cover a sufficiently large range of input and output pares. To do this, a long period of off-line training becomes imperative and this period may stretch to hundreds of seconds or even days [2-4] depending on specific characteristics of the controlled plants. Moreover, special control signals are sometimes required to excite the controlled plant. For these reasons, the indirect learning strategy is infeasible for most on-line control applications.

The direct control strategy can avoid this problem if a priori qualitative knowledge of the plant is available. Although this qualitative knowledge is not difficult to obtain in most control problems, our study shows that the strategy sometimes makes the actuator saturate, causing the output of the controlled plant unstable. As has been observed in the example provided later, the problem becomes even more serious when the delays of control actions are considered, as most industrial control actuators have output limits and cannot move from one action to another instantaneously.

If random disturbances are added to the plant-control system, the off-line training method is even more difficult to apply. In a broad sense, the process of control is to control the interaction between the plant and the environment. Considering the changeable surroundings, a plant can be seen as time-varying when either the plant dynamics changes or the disturbances vary. As a result, off-line training undertaken long before the need to provide actual control has no real adaptive features and is therefore difficult to apply to a wide range of real-time control problems.

The principal task of this article is to design an on-line trained neural controller which can cope with the complex interactions between a nonlinear plant and changeable surroundings without any off-line training process. In contrast to the various neural control schemes introduced above, the proposed method has the following features. (a) With a modest amount of qualitative knowledge about the controlled plant, the scheme needs no identification process. (b) The size of the training data set is carefully limited to provide balance between the time requirement for the on-line training and the need to provide sufficient understanding of the plant dynamics. (c) The network is trained continually and can therefore be used with processes having time-varying characteristics. (d) A fuzzy relationship between the plant behavior and the training targets is established to tackle the saturation problem associated with other direct control strategies. By using these techniques, the on-line adaptive control has been successfully achieved and the performance of the direct neural controller has been effectively improved.

In order to evaluate the performance of the proposed neural controller, results from a nonlinear ship course-keeping simulation are presented. For comparison, a conventional discrete PID controller is also designed to perform the same task under the same circumstances.

The New Neural Controller Configuration

The Direct Adaptive Control Structure

The proposed direct neural controller represents an extension of Saerens's work [5]. In particular, a limit function element is now considered, as shown in Fig. 1. The limit function element exists widely in industrial processes. Although it will bring about nonlinearity in control systems, its inclusion will increase the simulation realism. In Fig. 1 u_k^c is the output of the neural controller while u_k^p is the real control action generated by the plant actuator. The error function E^T is defined as

$$E^T = \frac{1}{2}\sum_q \left(y_{k,q}^T - y_{k,q}\right)^2 = \frac{1}{2}\sum_q e_{k,q}^2 , \tag{1}$$

where the subscript q denotes the number of distinct plant outputs, and $y_{k,q}^T$ and $y_{k,q}$ respectively represent the qth desired target and the qth actual plant output at time k. The particular control task to be investigated later corresponds to a single-input single-output problem, therefore q=1 is the only possibility and it will be omitted in the discussion below.

Within each time interval from k-1 to k, the backpropagation algorithm [6] is used to update the connective weights, in the neural network controller, according to the relation

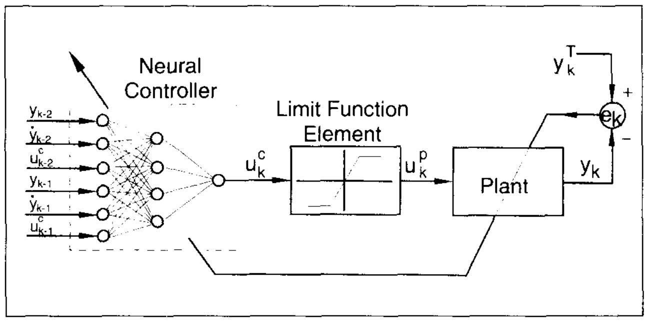


Fig. 1. Direct neural controller scheme.

$$w_{ij}(\tau+1) = w_{ij}(\tau) - \eta\frac{\partial E^T}{\partial w_{ij}(\tau)} + \alpha\Delta w_{ij}(\tau-1) , \tag{2}$$

where η is the learning rate, α is the momentum factor, and τ indicates the number of training iterations. A three-layer (input, hidden, and output) network is used for the neural controller (see Fig. 1). The hidden and output layers contain several processing units with an associated sigmoidal function

$$o_i = f(net_i) = \frac{1}{1+e^{-net_i}} , \tag{3}$$

subject to

$$net_i = \sum_j \left(w_{ij} \cdot o_j\right) + \theta_i . \tag{4}$$

Here o_i represents the output of units in the hidden and output layers, net_i is the summed input to the units in the hidden and output layers, and w_{ij} is the connective weight between input and hidden layers, or between hidden and output layers. θ_i is the threshold value for the units in the hidden and output layers.

There are two considerations in the use of the backpropagation network architecture—the right number of layers and the right number of units in a hidden layer. Cybenko [7] has shown that one hidden layer with sigmoidal function is sufficient to compute arbitrary decision boundaries for the outputs. Although a network with two hidden layers may give better approximation for some specific problems, de Villiers et al. [8] has demonstrated that networks with two hidden layers are more prone to fall into local minima. Furthermore, networks with two hidden layers need more CPU time than networks with one hidden layer. For these reasons, a network with a single hidden layer has been applied in this study.

Lippmann [9] has provided comprehensive geometrical arguments and reasoning to justify why the maximum number of units in a single hidden layer should equal M(N+1), where M is the number of output units and N is the number of input units. In practice, it is found that using fewer units is often sufficient. In our simulations, different numbers of hidden units (from two to seven) have been tested. It was found that a network with three to five hidden units often gave good results, and hence four units were used in the single hidden layer.

Using the chain rule, it follows that the required gradient of E^T, given in Equation (2), is determined using

$$\frac{\partial E^T}{\partial w_{ij}} = \frac{\partial E^T}{\partial u_k^p}\frac{\partial u_k^p}{\partial u_k^c}\frac{\partial u_k^c}{\partial w_{ij}} = \frac{\partial E^T}{\partial y_k}\frac{\partial y_k}{\partial u_k^p}\frac{\partial u_k^p}{\partial u_k^c}\frac{\partial u_k^c}{\partial w_{ij}} . \qquad (5)$$

In Equation (5) $\partial u_k^c / \partial w_{ij}$ can be readily obtained through the backpropagation algorithm [6], but the exact calculation of $\partial y_k / \partial u_k^p$ is difficult to determine because of the unknown plant dynamics. To overcome this problem, Psaltis et al. [1] presented two derivative approximations. By slightly changing each input to the plant at an operating point, and measuring the changes in the output, Psaltis et al. suggested

$$\frac{\partial y_k}{\partial u_k^p} \approx \frac{y_k\left(\bar{u} + \Delta u_k^p\right) - y_k(\bar{u})}{\Delta u_k^p} . \qquad (6)$$

Alternatively, by comparing the changes of the derivative related variables with values in the previous iteration, the derivative can be approximated using the relationship

$$\frac{\partial y_k}{\partial u_k^p} \approx \frac{y_k - y_{k-1}}{u_k^p - u_{k-1}^p} . \qquad (7)$$

However, it has been observed in earlier reported simulations [10] that the use of approximation (6) or (7) often causes ambiguity for network training when the controlled plant has large inertia or when disturbances are added. Ambiguity in training means that the neural controller establishes a "cause and effect" relationship, between input and output, which is contrary to what would be expected from a clear understanding of the situation being investigated. Therefore, the derivative $\partial y_k / \partial u_k^p$ is approximated by the ratio of the signs of the changes in y_k and u_k^p, which is consistent with the work of Saerens et al. [5]. The term $\partial u_k^p / \partial u_k^c$ is also replaced by its sign, so that Equation (5) now takes the form

$$\frac{\partial E^T}{\partial w_{ij}} = \frac{\partial E^T}{\partial u_k^p}\frac{\partial u_k^p}{\partial u_k^c}\frac{\partial u_k^c}{\partial w_{ij}} = \frac{\partial E^T}{\partial y_k}\frac{\partial y_k}{\partial u_k^p}\frac{\partial u_k^p}{\partial u_k^c}\frac{\partial u_k^c}{\partial w_{ij}}$$
$$\approx \left(y_k - y_k^T\right) \cdot sign\left(\frac{\partial y_k}{\partial u_k^p}\right) \cdot sign\left(\frac{\partial u_k^p}{\partial u_k^c}\right) \cdot \frac{\partial u_k^c}{\partial w_{ij}} , \qquad (8)$$

where $sign\left(\partial y_k / \partial u_k^p\right)$ is generally known once some qualitative knowledge of the plant is available. Clearly knowledge of how the control signals, u_k^p, influence the plant outputs, y_k, will provide the required sign information. For example, in ship maneuvering the qualitative knowledge concerning ship heading ($y_k = \psi_k$) and rudder angle ($u_k^p = \delta_k$; see Fig. 2 for the adopted sign convention) is appreciation of the fact that application of rudder to port will change the ship heading to the port (despite the fact that on some occasions the ship may continue to turn to starboard due to its large inertia or the existence of wind forces on the port side). Therefore,

$$\frac{\partial y_k}{\partial u_k^p} = \frac{\partial \Psi_k}{\partial \delta_k} < 0 , \qquad (9)$$

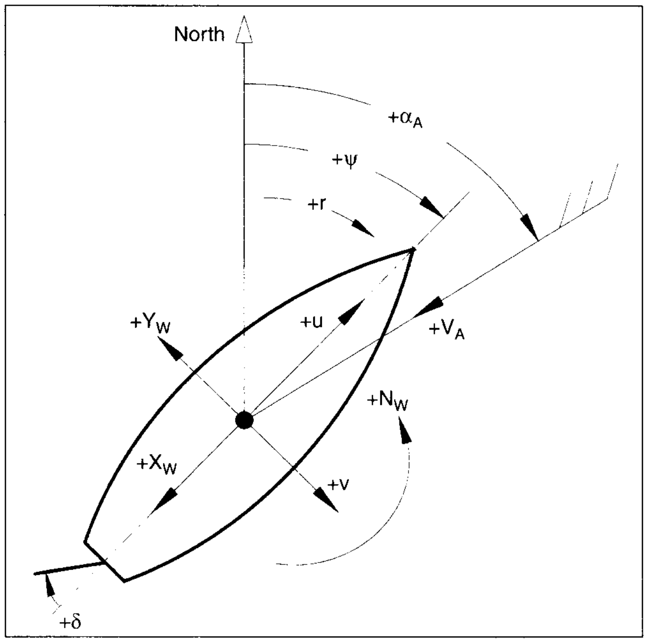


Fig. 2. Ship variable conventions.

or

$$sign\left(\partial y_k / \partial u_k^p\right) = -1 . \qquad (10)$$

Similarly, it is obvious in the marine context that an increase of u_k^c leads to an increase of u_k^p and therefore

$$sign\left(\partial u_k^p / \partial u_k^c\right) = 1 . \qquad (11)$$

Using Equation (8) with the given derivative signs provided in Equations (10) and (11), the neural controller will effectively output control signals with the correct direction according to the plant output error e_k.

Setting Interim Training Targets for the Neural Controller

Another problem identified for the earlier neural controller (Fig. 1) based simulations was the existence of situations in which the neural controller output u_k^c either continually switched between the maximum and minimum allowed values (*fluctuation*) or stuck to one of these values (*saturation*). At that time, network training was exclusively based upon the plant error e_k with the target fixed to the desired value y_k^T (refer to Equation (1)). Such a behavior of u_k^c has caused either slow responses or severe fluctuations in the plant output, especially when the plant was under external disturbances or noise. Furthermore, in most industrial process control problems the actuators are not such that they can switch from one output control signal to another instantaneously. For a hydraulic rudder, for example, the maximum rudder rate $\dot{\delta}_{\max}$ is 2.5deg/sec, which means it needs 24 seconds for a full starboard rudder (-30deg) to reach full port side (+30deg). It is not easy for the neural network to learn such characteristics.

Table 1. Fuzzy Rules for Selecting Interim Targets

		e_k						
		NB	NM	NS	ZR	PS	PM	PB
$\dot{e}_k$	NB	+B	+B	+B	+M	+M	+S	ZR
	NM	+B	+B	+M	+M	+S	ZR	-S
	NS	+B	+M	+M	+S	ZR	-S	-M
	ZR	+M	+M	+S	ZR	-S	-M	-M
	PS	+M	+S	ZR	-S	-M	-M	-B
	PM	+S	ZR	-S	-M	-M	-B	-B
	PB	ZR	-S	-M	-M	-B	-B	-B

PB = Positive Big, PM = Positive Medium, PS = Positive Small, ZR = Zero, NB = Negative Big, NM = Negative Medium, NS = Negative Small.

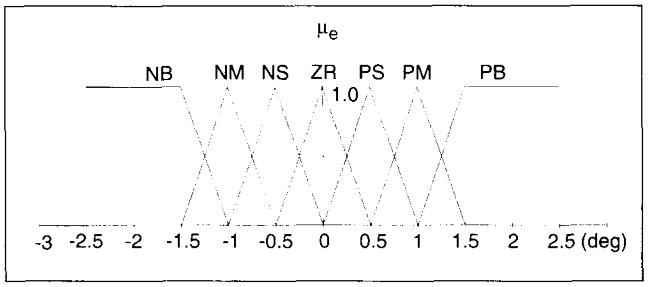


Fig. 3. Membership function for heading error e_k.

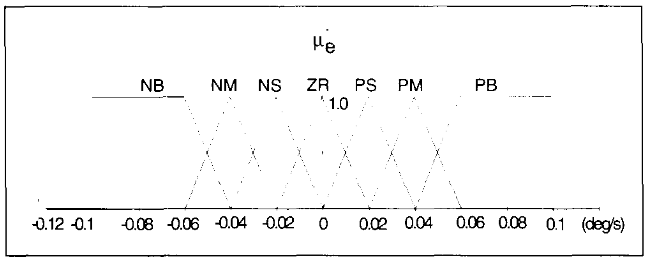


Fig. 4. Membership function for heading error change $\dot{e}_k$.

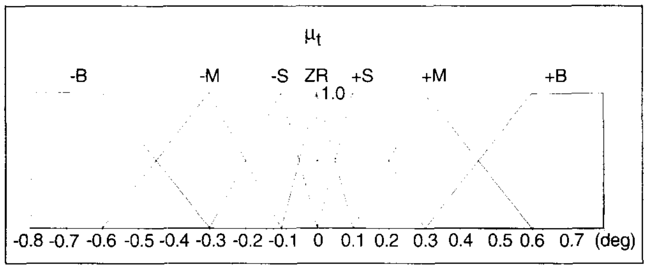


Fig. 5. Membership function for interim targets.

In order to improve the performance of the direct neural controller, a series of *interim targets* have been used instead of the final target. Selecting an interim target y_k^I rather than a static final target y_k^T (i.e., the desired plant output) is more reasonable for network training, because there is no sense in setting a target which the neural controller is unable to achieve within one operation interval (1 second in this article). The underlying idea is to "slice" the final big target into several small reachable interim targets which are generated from a set of fuzzy rules given by Table 1.

Unlike fuzzy logic controllers, Table 1 does not present a control strategy. Instead, it provides a reachable training target for the neural controller so that the fluctuation and the saturation problems referred to earlier can be effectively tackled. The interim targets obtained from Table 1 provides the controller with the information "where the plant should go" but not "how to reach there." In order to know "how to reach" the interim targets, a mechanism is required to infer how strong the control actions should be according to the interactions between the plant and the environment. This is the task of the neural network controller. The basic idea behind the neural network approach is to capture the system input-output relations and then to generate an appropriate control signal to drive the plant to an identified interim target.

The reader should note that the interim targets -B through zero to +B, listed in Table 1, indicate the size of the change to be made to the current plant state y_k and should therefore be added to y_k. For instance, assuming the error e_k is negative big (NB) and its change $\dot{e}_k$ is also negative big (NB), the interim target y_k^I should be ($y_k + B$). The strategy presented in Table 1 is straightforward and has general adaptability. Obviously it will need some adjustment if a rather different plant is to be controlled, for example, a 20T boat rather than a 20,000T ship. The fuzzy notation values, NB to PB, are also determined by the particular characteristics of the controlled plant. For simplicity of computation, the membership functions for the heading error e_k (μ_e), the heading error change $\dot{e}_k$ ($\mu_{\dot{e}}$), and the interim targets (μ_t) are defined by piecewise straight lines. Hence, when notation values (NB to PB) are given, the shapes of the membership functions will be decided. To see the sensitivity of the neural controller to the notation values, different NB to PB values for e_k and $\dot{e}_k$ with different shapes of the membership functions (μ_e and $\mu_{\dot{e}}$) have been simulated. Figs. 3 to 5 provide one of the choices. Other tested values will be given in the Evaluation Section.

The interim target is obtained through the following process. Depending upon the values of e_k and $\dot{e}_k$, the possible choices of values of μ_e and $\mu_{\dot{e}}$ are identified from Figs. 3 and 4, respectively. These, together with the selected fuzzy rules of Table 1, identify the region of μ_t defined in Fig. 5. With the inference and aggregation processes complete, the defuzzification is undertaken basing on Fig. 5 to determine the definite value of the interim target. To do this, the discrete centroid method is used with a discrete unit step of 0.01 expressed in the form

$$\text{interim target} = \frac{\sum_{i=-0.8}^{0.8} i \cdot \mu_t^i}{\sum_{i=-0.8}^{0.8} \mu_t^i}. \qquad (12)$$

Here the limits of summation define the minimum and maximum allowable values identified from a knowledge of the characteristic of dynamics of the plant. For details of the fuzzy

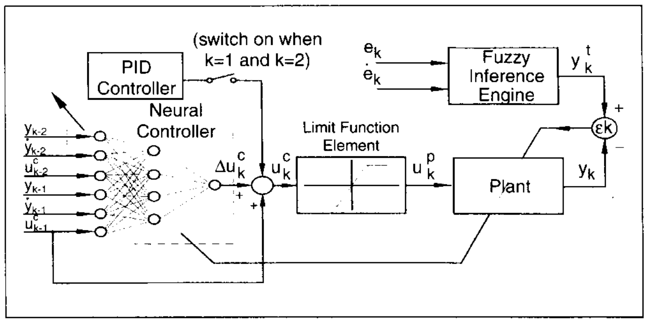


Fig. 6. Modified direct neural controller.

inference, aggregation, and defuzzification processes, readers may refer to Yamakawa's paper [11].

Because the control signal u^c_k is obtained by processing e_k and $\dot{e}_k$ using the fuzzy table and then by training the neural network, the high-frequency control signals are significantly reduced in number in a noisy environment. The positive outcome from this is that the neural controller can filter out the influence of most of the measurement noise during the control process. This will be demonstrated in simulation results later.

Together with the use of interim targets, the following method is applied in order to reduce abrupt changes in the control signal, u^c_k. During each time interval the output range of the neural network (defined as Δu^c_k) is scaled within the actuator's maximum range of ability. In the ship-steering problem, the rudder maximum range for a 1 second interval is from 0°/*s* to $\dot{\delta}_{\max}$ (2.5°/*s*). This defines $\Delta u^c_k \in (0°, 2.5°)$. The output signal of the neural controller at the time *k* is then

$$u^c_k = u^c_{k-1} + \Delta u^c_k \, . \tag{13}$$

With the above adjustment, the modified direct neural controller scheme is presented in Fig. 6. The associated interim targets y^t_k are generated with the use of the fuzzy inference engine, as described earlier. Accordingly, the quantities E^T and y^T_k in Equation (8) are replaced by E^t and y^t_k, respectively, with the new error function defined by

$$E^t = \frac{1}{2}\left(y^t_k - y_k\right)^2 = \frac{1}{2}\varepsilon^2_k \, . \tag{14}$$

For the purpose of better training, the best number of inputs to the network (see Fig. 6) may not be six. Earlier plant behaviors (for example, for periods *k-3*, *k-4*, etc.) might be useful, but this will increase the CPU time for network training and hence for this simulation only six inputs are used.

A Brief Description of a Nonlinear Ship Model

The ship model simulated in this study is the Hydronautics model of a Mariner hull form from Goodman et al. [12]. In particular the surge, sway, and yaw equations of the ship motions are assumed to have the following nonlinear forms:

$$\begin{aligned}
&m\left(\dot{u} - vr - x_G r^2\right) = \\
&+\frac{\rho}{2}L^4\left(X_{rr}' r^2\right) + \frac{\rho}{2}L^3\left(X_{\dot{u}}'\dot{u} + X_{vr}' vr\right) + \frac{\rho}{2}L^2\left(X_{vv}' v^2\right) \\
&+\frac{\rho}{2}L^2u^2\left(a_1 + a_2\eta + a_3\eta^2\right) + \frac{\rho}{2}L^2u^2\left(X_{\delta\delta}'\delta^2 + X_{\delta\delta\eta\eta}'\delta^2\eta^2\right) \\
&+\frac{\rho}{2}L^2\left(X_{vv\eta}v^2\right)(\eta - 1) \, ,
\end{aligned} \tag{15}$$

$$\begin{aligned}
&m(\dot{v} - ur - x_G\dot{r}) = \\
&+\frac{\rho}{2}L^4\left(Y_{\dot{r}}'\dot{r} + Y_{r|r|}' r|r|\right) + \frac{\rho}{2}L^3\left(Y_{\dot{v}}'\dot{v}\right) \\
&+\frac{\rho}{2}L^3\left(Y_r' ur + Y_{|r|\delta}' u|r|\delta + Y_{v|r|}' v|r|\right) \\
&+\frac{\rho}{2}L^2\left(Y_\delta' u^2\delta\right) + \frac{\rho}{2}L^3Y_{r\eta}' ur(\eta - 1) \\
&+\frac{\rho}{2}L^2\left(Y_*' u^2 + Y_v' uv + Y_{v|v|}' v|v|\right) \\
&+\frac{\rho}{2}L^2\left(Y_{*\eta}' u^2 + Y_{v\eta}' uv + Y_{v|v|\eta}' v|v| + Y_{\delta\eta}' u^2\delta\right)(\eta - 1) \, ,
\end{aligned} \tag{16}$$

$$\begin{aligned}
&I_z\dot{r} + mx_G(\dot{v} + ur) = \\
&\frac{\rho}{2}L^5\left(N_{\dot{r}}'\dot{r} + N_{r|r|}' r|r|\right) + \frac{\rho}{2}L^4\left(N_{\dot{v}}'\dot{v}\right) \\
&+\frac{\rho}{2}L^4\left(N_r' ur + N_{|r|\delta}' u|r|\delta + N_{v|r|}' v|r|\right) \\
&+\frac{\rho}{2}L^3\left(N_*' u^2 + N_v' uv + N_{v|v|}' v|v|\right) \\
&+\frac{\rho}{2}L^3\left(N_\delta' u^2\delta\right) + \frac{\rho}{2}L^4N_{r\eta}' ur(\eta - 1) \\
&+\frac{\rho}{2}L^3\left(N_{*\eta}' u^2 + N_{v\eta}' uv + N_{v|v|\eta}' v|v| + N_{\delta\eta}' u^2\delta\right)(\eta - 1) \, ,
\end{aligned} \tag{17}$$

where *m* and I_z are respectively the mass and the moment of inertia of the ship, x_G is the longitudinal position of the center of gravity with respect to origin of selected reference system, ρ is the density of the seawater, *L* is the ship length, $\eta = u_c/u$, where u_c is the command speed, a_i (*i*=1, 2, 3) are the propulsion influence coefficients for the Mariner, δ is the control signal (rudder angle), and *u*, *v*, and *r* are forward speed, lateral speed, and heading rate of the ship, respectively. The physical quantity η in Equations (15) to (17) should not be confused with the learning rate η in Equation (2). The non-dimensionalized hydrodynamic derivatives (X_{rr}', $X_{\delta\delta\eta\eta}'$, $Y_{\dot{r}}'$, $Y_{r|r|}'$, N_δ', $N_{*\eta}$, ...) are assigned the fixed values provided by Goodman et al. [12]. Inclusion of the wind forces and moment requires that extra terms, X_W, Y_W, and N_W, (see Fig. 1 for the sign conventions), be added respectively to the right-hand side of the Equations (15) to (17), and to do this Isherwood's work [13] is used. Readers may refer to [12,13] for details of these parameters.

In the absence of a real ship with onboard real time collection of δ, *r* and the heading ψ ($\psi = \int_0^t r dt$) values, the mathematical nonlinear ship model outlined above is used to provide these values for control simulation. Nevertheless, if neural networks can be trained to control the ship model described by Equations (15) to (17), then in principle the methodology should be transferable to real ships.

The simulation task is to find a controller to regulate the ship heading ψ_k to some desired course ψ_k^T subject to the input constraints:

- rudder angle δ: from -15.0° to +15.0°
- rudder rate $\dot{\delta}$: from -2.5°s^{-1} to +2.5°s^{-1}

Constraining the rudder angle within the stated narrow band increased the difficulty of the control problem.

The heading error and its rate have the following ranges:

$e_k = \psi_k^T - \psi_k$: from -25.0° to +25.0°

$\dot{e}_k = \psi_{k-1} - \psi_k$: from -0.75°$s^{-1}$ to +0.75°s^{-1}

To make the simulation more realistic, maneuvers with random wind forces and random measurement noise are considered subject to:

- wind speed V_A: from 0 m/s to 40 m/s (random)
- wind direction α_A: from -90° to +90° (random)
- measurement noise: 20% · R_k · e_k (R_k is a set of random sequence from -1 to +1)

Evaluation

To evaluate the control quality of the proposed neural controller, the nonlinear ship model, Equations (15) to (17), is simulated for course-keeping control. The target course is 10° ∈ [0s, 200s), then -10° ∈ [200s, 400s), and finally 10° ∈ [400s, 600s]. Since the Mariner ship dynamics is relatively slow, Equations (15) to (17) are integrated using a finite difference scheme with a time step of 0.01 second. The time interval for each control action is 1 second.

At the beginning of each time interval k, the weights and thresholds of the neural network are initialized to small random values uniformly distributed between -0.05 and 0.05. Then the previously measured ship heading ψ, turning rate r, and rudder angle δ at the time k-1 and k-2 are taken from memory to form the training data. In other words, the input data set for network training consists of six variables, specifically $\psi_{k-1}, r_{k-1}, \delta_{k-1}^c, \psi_{k-2}, r_{k-2}, \delta_{k-2}^c$. The role of the network training is to find an appropriate control signal δ_k^c so that the squared error between ψ_k^t and ψ_k (plant output) can be minimized. During the first few control cycles (k=1 and k=2), when the six data values above are not available, a PID controller is switched on. Since the network should provide an appropriate control signal δ_k^c before the next time interval comes, the network training "finishes" when

$$E^t = \frac{1}{2}\left(y_k^t - y_k\right)^2 \le 0.0001, \tag{18}$$

or terminates when

$$\tau_{\max} = 4000, \tag{19}$$

where $\tau_{\max}$ is the maximum number of training iterations.

Table 2. Different Combinations of Notation

	e_k	$\dot{e}_k$
Case 1	tight	tight
Case 2	tight	loose
Case 3	loose	tight
Case 4	loose	loose

Table 3. The Notation Values for e_k and $\dot{e}_k$

		NB	NM	NS	ZR	PS	PM	PB
e_k (deg)	tight	-1.5	-1.0	-0.5	0.0	0.5	1.0	1.5
	loose	-3.0	-2.0	-1.0	0.0	1.0	2.0	3.0
$\dot{e}_k$ (deg/s)	tight	-0.06	-0.04	-0.02	0.0	0.02	0.04	0.06
	loose	-0.12	-0.08	-0.04	0.0	0.04	0.08	0.12

The simplicity of the data set and the training strategy discussed above have effectively guaranteed on-line training and real-time control. In most simulation tests on a workstation, training on each data point can be finished within 0.7 second with a learning rate of η=0.7 and a momentum factor of α=0.7. This compares favorably with the control interval of 1 second. In some cases, local minima have been observed during the network training and the network did not converge to satisfy condition (18) above. Under such circumstances, the network training is "forced" to terminate using Equation (19). It is obvious that the neural controller may generate wrong control signals when local minima occur. Fortunately, this problem is not as serious as one would imagine for the system stability, which is due to the following facts: (a) When the neural controller ends up with a local minimum, the wrong control process lasts for only one second, during which the largest possible change in the control actuator (rudder angle δ_k^p) is 2.5°; (b) This wrong control signal will not stay there and will be replaced during the next control cycle because the same training process is repeated during each time interval. Hence, the performance of the neural controller will not be much influenced by the occasional lack of convergence. In Figs. 9(b) and 11(b), the few small blips on the rudder curves illustrate the presence of local minima during the neural network training.

Different NB to PB values have been tested. The simulation results presented in each group include four different curves corresponding to four cases with different combinations of the notation values, as shown in Table 2. The "loose" and "tight" values for e_k and $\dot{e}_k$ are given in Table 3.

A PID controller was also designed to perform the same course-keeping task for comparison. For continuous time signals, the time-domain relation between the input and output of the PID controller is

$$\delta(t) = K_p e(t) + K_I \int_0^t e(t)dt + K_D \dot{e}(t). \tag{20}$$

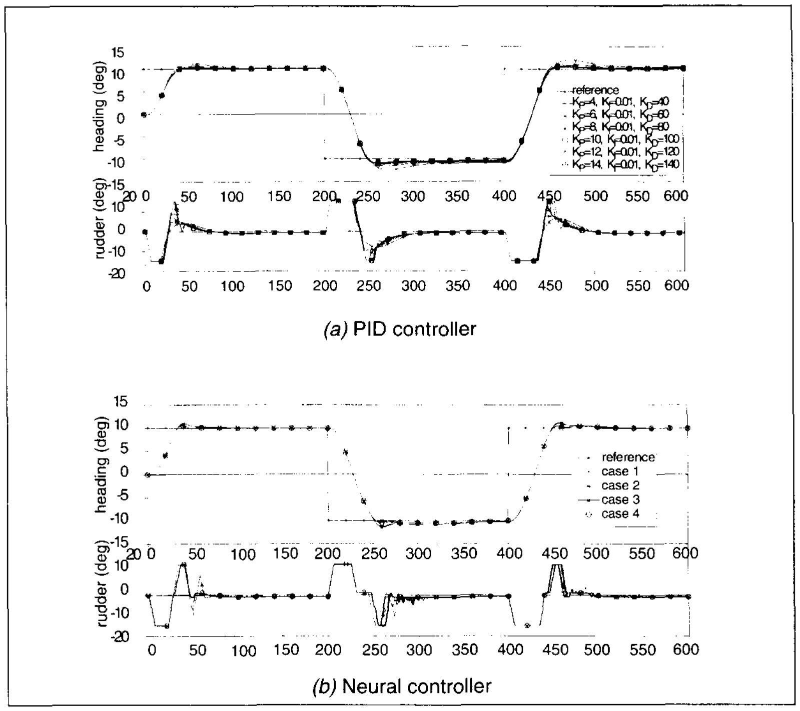


Fig. 7. Course-keeping under perfect condition.

For the discrete time signals, a possible PID control algorithm has the following form [14]

$$\delta_k = \delta_{k-1} + (K_P + K_I + K_D)e_k - (K_P + 2K_D)e_{k-1} + K_D e_{k-2}, \quad (21)$$

where e_k is the difference between measured and desired heading:

$$e_k = \Delta\psi_k = \psi_k^T - \psi_k \,. \quad (22)$$

The difficulty in the design of the PID controller is that plant control is facilitated through measurement of the quantities e_k, e_{k-1}, and e_{k-2}. But the setting of appropriate parameters K_P, K_I, and K_D according to different situations requires other techniques or human experience. A proper choice of the K_P, K_I, and K_D values can make the PID controller perform quite well in a specific case, but for cases where internal and external conditions change (such as wind disturbances) the PID controller cannot guarantee good performance. To have an overview of the PID controller performance, six different combinations of K_P, K_I, and K_D have been used in simulations:

K_P = (4, 6, 8, 10, 12, 14),

K_I = 0.01, and

K_D = (40, 60, 80, 100, 120, 140).

The simulation tests consist of course-keeping under various conditions. To compare the neural controller performance with the PID controller, the error between the desired and achieved heading is evaluated for each case.

No Disturbance, No Noise

The first simulation test compared the performance of the neural and PID controllers under ideal conditions of no wind disturbances and no measurement noise. The results are shown in Fig. 7. As expected, under perfect conditions the PID controller achieved smooth responses with an acceptable overshoot. Although the heading overshoot can be reduced further by increasing the proportional gain (K_P) and the differential gain (K_D), this is not feasible when noise is added. The neural controller also exhibits good performance with even smaller overshoot. Different notation values (NB, NM, ...) do not have much influence upon the controller's performance, although slightly more adjustments have been observed when notation values for both e_k and $\dot{e}_k$ tend to be tight (see Case 1 in Fig. 7(b)).

Wind Disturbance, No Measurement Noise

The second simulation test was similar to the first but included a random wind disturbance. The wind speed, V_A, changes randomly every five seconds from 0m/s to 40m/s, while its direction, α_A, changes every 25 seconds from -90° to +90°. Fig. 8 shows the normalized wind speed and defines wind direction as applied in the simulations.

The simulation results for this case are given in Fig. 9. It can be observed that while the PID controller is obviously influenced by the wind disturbance, the neural controller tracks the desired course rather well.

Wind Disturbance Plus Measurement Noise

The next simulation test was identical to the previous test with the addition of measurement noise, the magnitude of which is shown in Fig. 10. All other test conditions remain the same as before, and Fig. 11 shows the results. It can clearly be seen that the measurement noise has an obvious negative effect on the PID controller with both the ship heading and the rudder angle being corrupted, especially when K_P and K_D take larger values. But the neural controller retains almost the same good performance as in the previous test (Fig. 9(b)). Comparing the rudder movement shown in Fig. 11(a) and 11(b), the noise-resistance of the proposed scheme is illustrated. This feature is due to the following fact. Although the noise may slightly change the choice of interim targets obtained from Figs. 3, 4, and 5 and Table 1, it is not directly linked to the control actions and hence it will not cause radical changes in control signals generated by the neural controller.

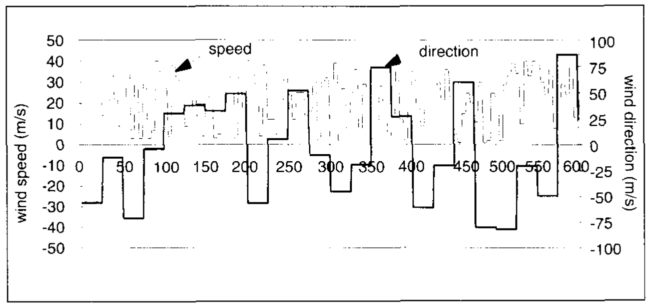


Fig. 8. Wind speed and its direction.

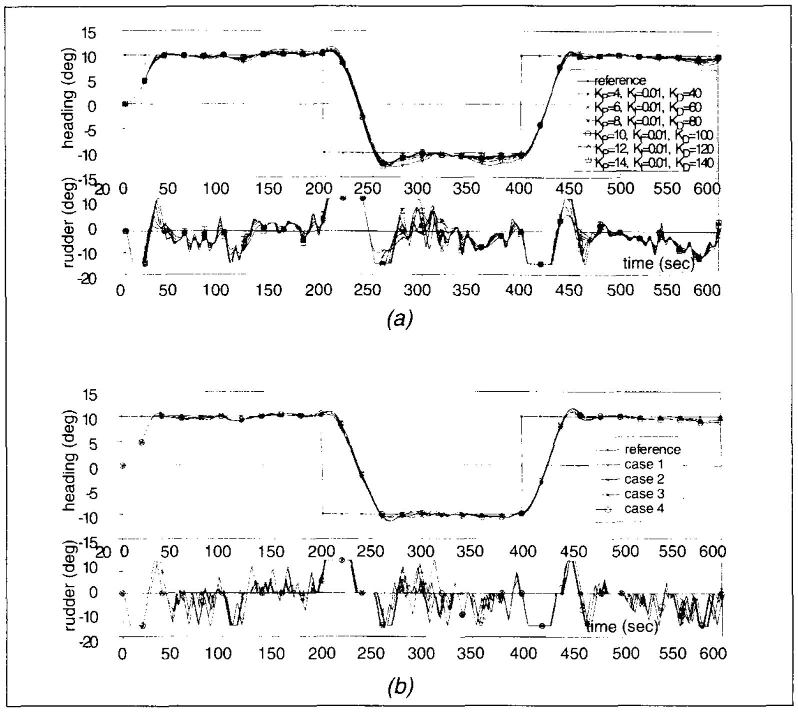


Fig. 9. Course-keeping under wind disturbance.

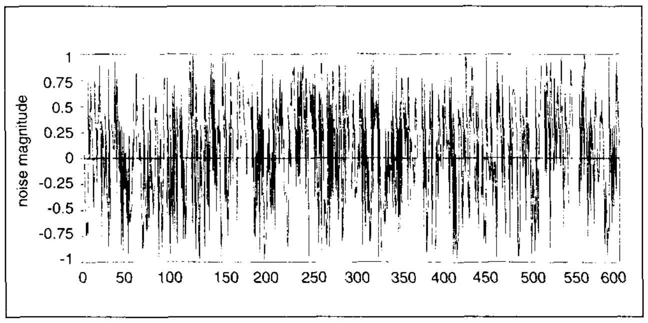


Fig. 10. Pseudo-random measurement noise sequence R_k.

Conclusion

This article describes a neural network approach for industrial tracking control. In order to overcome the problems associated with the direct neural controller architecture, a fuzzy relation between the plant behavior and the interim training targets is set up. The merit of this strategy lies in the adaptive property it imparts to the controller to cope with complicated interactions between the nonlinear time-varying plant and the changeable surroundings. Furthermore, the careful choice of neural network structure and the training data size has made on-line control possible. Results from ship course-keeping control simulations demonstrate the feasibility of the proposed neural controller when compared to the conventional PID controller, especially in the presence of external disturbances and measurement noise.

From the simulation results, it is also noticed that the neural controller shows more control adjustments than that of the PID controller. However, this need not cause very serious increase in wear and tear because of the constraints on rudder angle and rudder rate. In any event, to get better control performance under severe wind disturbance and measurement noise, more adjustments are necessary. Nevertheless, to reduce unnecessary movements of the rudder angle without influencing the control quality significantly, some techniques such as including control terms in the energy function (14) or adjusting the shape of the membership functions might be useful.

It should also be noted that the values in Figs. 3, 4, and 5 and Table 3 are set up for the specific ship model applied in this article. For different plants, some testing would be needed to decide appropriate values for good control quality. Nevertheless, the simplicity of the neural controller scheme and its practicality for real-time control provide a new approach for implementing neural network applications for a variety of on-line industrial tracking control problems.

The neural controller discussed in this article has been restricted to a single-input single-output process. Extending this to a general multivariable case (such as harbor approaches using both rudder and propeller) introduces some additional problems, such as the more complex neural network structures, the increase of training time, and other possible adverse conditions for on-line training. These related issues will be investigated in our future research.

References

[1] D. Psaltis, A. Sideris, and A.A. Yamamura, "A Multilayered Neural Network Controller," *IEEE Control Systems Magazine*, vol. 8, no. 2, pp. 17-21, April 1988.

[2] J. Tamomaru and S. Omatu, "Process Control by On-Line Trained Neural Controllers," *IEEE Trans. Industrial Electronics*, vol. 39, no. 6, pp. 511-521, Dec. 1992.

[3] D. Nguyen and B. Widrow, "Neural Networks for Self-Learning Control Systems," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 18-23, April 1990.

[4] Q.H. Wu, B.W. Hogg, and G.W. Irwin, "A Neural Network Regulator for Turbogenerators," *IEEE Trans. Neural Networks*, vol. 3, no. 1, pp. 95-100, Jan. 1992.

[5] M. Saerens and A. Soquet, "A Neural Controller," in *Proc. First IEE International Conference on Artificial Neural Networks*, London, pp. 211-215, Oct. 1989.

[6] D. E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, vol. 1, D.E. Rumelhart, and J.L. McClelland, eds. Cambridge, MA: MIT Press, 1986.

[7] G. Cybenko, "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Controls, Signals and Systems*, vol. 2, no. 4, pp. 303-314, 1989.

[8] J. de Villiers, E. Barnard, "Backpropagation Neural Nets with One and Two Hidden Layers," *IEEE Trans. Neural Networks*, vol. 4, no. 1, pp. 136-141, Jan. 1993.

[9] R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE Acoustics, Speech, and Signal Processing Magazine*, pp. 4-22, April 1987.

[10] G.E. Hearn, P. Sen, and Y. Zhang, "Ship Motion Control by On-Line Trained Neural Networks," in *Proc. 3rd International Conference on Maneuvering and Control of Marine Craft*, Southampton, U.K., pp. 75-88, Sept. 1994.

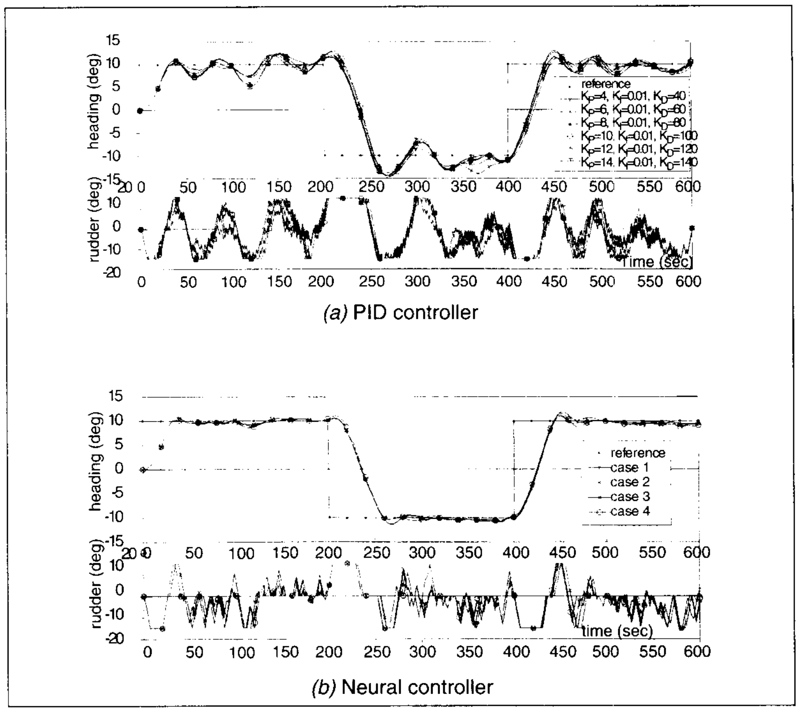


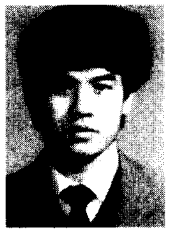
Fig. 11. Course-keeping under wind disturbance and measurement noise.

[11] T. Yamakawa, "A Fuzzy Inference Engine in Nonlinear Analog Mode and its Application to a Fuzzy Logic Control," *IEEE Trans. Neural Networks*, vol. 4, no. 3, pp. 496-510, May 1993.

[12] A. Goodman, M. Gertler, and R. Kohl, "Experimental Techniques and Methods of Analysis Used at Hydronautics for Surface Ship Maneuvering Predictions," in *Proc. 11th ONR Symposium on Naval Hydrodynamics*, London, pp. 55-113, 1976.

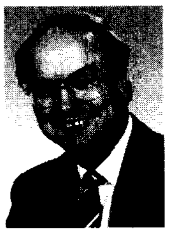
[13] J.W. Isherwood, "Wind Resistance of Merchant Ships," *Transactions of Royal Institution of Naval Architects*, vol. 115, pp. 327-335, 1973.

[14] J. Van de Vegte, *Feedback Control Systems*, 2nd ed., Englewood Cliffs, N.J.: Prentice-Hall, Inc., Chapter 9, pp. 260-263, 1990.



Yao Zhang was born in Tianjin, the People's Republic of China, in 1960. He received the B.Sc. and M.Sc. degrees in Marine Technology in 1983 and 1986, respectively, from Dalian Maritime University (DMU), Dalian, P.R. China. From 1986 to 1992, he worked in the Department of Navigation at DMU as a lecturer (1988) and later as an associate professor (1991). Currently he is completing his Ph.D. research work in marine technology at the University of Newcastle upon Tyne, U.K. His current research interests include dynamical system identification and control by neural networks and the applications of neural networks to marine problems.

Pratyush Sen is a senior lecturer in marine technology and associate director of the Engineering Design Center at the University of Newcastle upon Tyne, U.K. He graduated in naval architecture from the Indian Institute of Technology at Kharagpur, India, in 1971. In 1972 he obtained a Norwegian Agency for International Development (NORAD) scholarship to pursue doctoral studies in Norway and received his doctorate in naval architecture from the University of Trondheim in 1975. His current research interests include multiple-criteria decision making, knowledge-based design systems, design for safety, ship control using neural networks, parallel processing techniques in ship voyage management, and design and analysis of marine transport systems. He is the author of numerous papers in international journals and is a member of several professional and international committees.



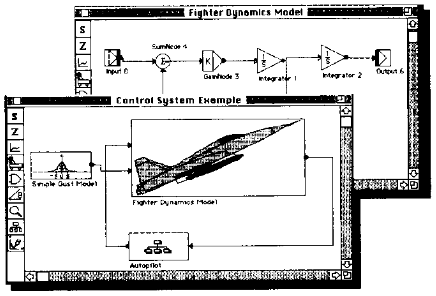
Grant E. Hearn is professor of hydrodynamics, head of the Department of Marine Technology and the center coordinator for marine-technology-related research at Newcastle University. He graduated in industrial mathematics from the Bath University of Technology in 1970 and gained his M.Sc. in applied mathematics from Sheffield University in 1971. Following periods as a research fellow in mathematics at Sheffield University and Naval Architecture at Strathclyde University, he joined the British Ship Research Association in 1976. There he was quickly promoted to head of the Mathematics Group. He joined the University of Newcastle as a lecturer in 1978. His industrial experience includes working as a mathematician in aircraft flight and control, optical telecommunication systems, and glass technology. His current research interests include hydrodynamics, seakeeping and maneuvering, and design and the applications of neural networks and genetic algorithms to ship design and control problems. He is the research manager for the U.K. ship maneuverability program based at Newcastle, Glasgow, and Southampton Universities.

Just √Build Simulate Plot Analyze it **Better!**

BuildSim™ 1.5

for PC/Windows and the Macintosh®

Unbelievable Capabilities!! Unbelievable Price!!



New Features

- Hierarchical System Blocks
- Frequency Response
- Custom Code* (C/FORTRAN)
- Fast Fourier Transforms
- Text Equation Block
- Add Text and Graphics
- Virtually Unlimited Model Size
- Integrated Plotter
- So Much More!

Nonlinear Dynamic Simulation
filters - mechanical systems - valves
signal processing - aerodynamics - economics

More Blocks
hierarchical systems - custom C/FORTRAN*
text equations - lookup tables - stochastics
signal sources - transcendentals
dynamics - logic

Same Great Price!
159^{95}

30-day, no-risk guarantee! CALL NOW,
1-800-819-8819

plus shipping & handling
MC / VISA

MICROSOFT® WINDOWS™ COMPATIBLE

Tangent Systems
P.O. Box 83338 San Diego, CA 92138-3338 (619) 222-3554
Albuquerque, NM (505) 298-4987

*Available on PC/Windows only.

BuildSim is a trademark of Tangent Systems; other product and brand names are trademarks or registered trademarks of their respective holders.

Reader Service Number 4