Applied Artificial Intelligence, 18:447–466, 2004
Copyright © Taylor & Francis Inc.
ISSN: 0883-9514 print/1087-6545 online
DOI: 10.1080/08839510490442076

Taylor & Francis
Taylor & Francis Group

STABILIZATION OF THERMAL NEUROCONTROLLERS

GERARDO DÍAZ
Modine Manufacturing Co., Racine, WI, USA

MIHIR SEN, K.T. YANG, and
RODNEY L. McCLAIN
Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN, USA

This work deals with the stabilization of neurocontrollers used in thermal applications. The control system can be reduced to an iterative, nonlinear map in time, and its linearization enables a stability analysis. For simple neural networks with few neurons, the eigenvalues can be analytically calculated in terms of the synaptic weights and biases. However, unless care is taken, usual training methods can drive the network to weights and biases such that the corresponding control system is unstable. A modified backpropagation training method is developed here to simultaneously minimize the target error and increase the dynamic stability of the system. Numerical computations are used to analyze the stability of realistic neural networks and their corresponding control systems. The techniques developed are used on an experimental heat-exchanger facility where the stability results are tested and validated.

A complex thermo-hydraulic system can be defined as one which, although containing subsystems whose behavior can be predicted by mathematical models, is itself too complicated for that. These systems are difficult to predict from first principles, using the basic governing equations of mass, momentum, and energy in differential form. Even though the use of computational fluid dynamics is commonplace, currently available hardware and methodologies do not permit solutions in more than in fairly simple geometries and/or for laminar flows. For flows under realistic conditions, such as in a heat exchanger, one can resort to simplifications like one-dimensionality and empirically obtained heat transfer coefficients. The net result is that whatever inaccuracies that are in these assumptions are reflected in the

We gratefully acknowledge the support of the late Mr. D. K. Dorini of BRDGTNDR for this and related projects in the Hydronics Laboratory. G.D. also thanks the Organization of American States for a PRA Fellowship.

Address correspondence to Mihir Sen, Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN 46556, USA. E-mail: Mihir.Sen.1@nd.edu

results. In addition, the characteristics of a working heat exchanger change considerably over time due to fouling so that predictions, however good in the beginning, eventually develop large errors. Artificial neural networks (ANNs) have been suggested as a way of predicting heat exchanger performance under both steady-state (Díaz et al. 1999; Sen and Yang 2000; Pacheco-Vega et al. 2001a; 2001b) and dynamic (Díaz et al. 2001a) conditions to overcome these problems. They can model nonlinear behavior and can be re-trained continuously or as needed.

The control of the temperature of a fluid exiting a heat exchanger is of importance in many thermal processes for manufacturing, climate control, and other applications. ANNs have also been used for this purpose by many authors. Ramaswamy et al. (1995), Bittanti and Piroddi (1997), and Dasgupta et al. (2001) have directly used trained neural networks for the temperature control of heat exchangers. Díaz et al. (2001a; 2001b) demonstrated the ability of this type of control system to adapt to changing circumstances. Others who have reported the use of neural network-based control of heat exchangers include AlDuwaish and Karim (1996), who combined a feedforward multilayer neural network and an auto-regressive moving average linear model; Matko et al. (1998) who proposed the use of a nonlinear autoregressive method; Riverol and Napolitan (2000) who used it to tune a PID controller; and Quek and Wahab (2000) who addressed the larger issue of integrated process supervision for the real-time control of an industrial heat-exchanger process.

The most common neural network configuration for this purpose is that of a multi-layer, feedforward ANN. Each layer has a number of neurons that are interconnected by means of synaptic weights, and each neuron has a bias. An activation function (here a logistics function) relates the output to the input of a neuron. Backpropagation (Rumelhart et al. 1986) and gradient descent (Pierre 1986) are among the methods that are commonly used during training to determine the weights and biases. During this process, the ANN learns the relationship between a given set of input–output data. For use in control systems, the steady-state ANN can be extended to predict dynamic processes in which the variables change with time. Such a thermal neurocontroller was experimentally tested in a heat exchanger test facility (Díaz et al. 2001a; 2001b). Though the controller functioned properly, there was no assurance that it always would from the point of view of stability. A control system has to be stable and, from a practical perspective, this is especially important for a system that is allowed to change over time, like an adaptive controller based on ANNs. The digital control system, as will be shown later, can be represented by a nonlinear map between previous instants in time and the present. To achieve control, the set point should be a fixed point of the map that is locally attracting. This is a necessary condition but not sufficient, since there may be other stable fixed points that are also locally attracting.

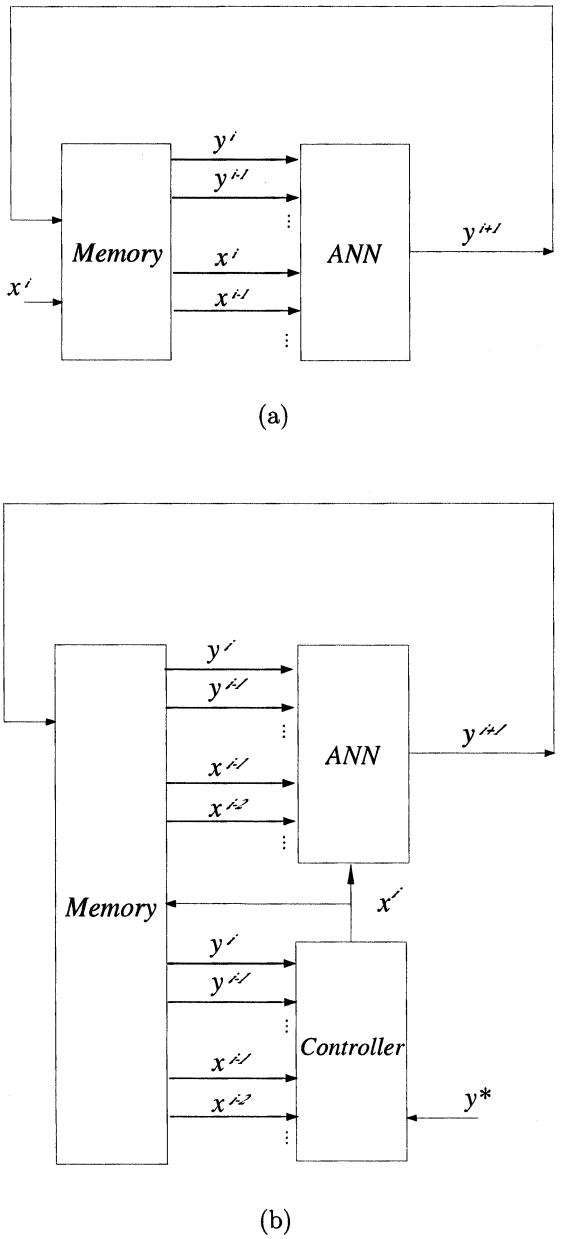
Furthermore, unlike its linear counterpart, the stability characteristics of a nonlinear control system vary over the operating range of the controller. For these reasons, it is important that there be some protection against instability built into the training of the neural network while it works to minimize the target error.

Some stability analyses of neurocontrollers have been performed by previous authors. Jin at al. (1993) studied the training problem in discrete-time dynamic neural networks, following the dynamic backpropagation algorithm of Pineda (1987) by modification of the learning rate. Hrycej (1995) looked at the existence of fixed points in neurocontrollers representing nonlinear differential systems and analyzed the effect of feedback and feed-forward on the control system. Delgado (1998) used a describing function technique to present the stability analysis of closed-loops systems with a linear plant and a neurocontroller. Taking a different perspective on the problem, we will consider the control system as an iterated map. To understand the relation between the behavior of the system and its characteristics, we will study some simple networks first. A training technique that incorporates stabilization along with error minimization will be developed, and the method will be tested on the control of the air temperature coming out of a heat exchanger.

NEUROCONTROLLER AS ITERATED MAP

For a system in a steady-state, the input and output variables are time-invariant. In a digitally controlled time-dependent system, however, the variables are functions of time which are sampled periodically at a constant time interval Δt. Here we are interested in a control methodology that brings the single-input-single-output thermal system to a given steady state. We are thus interested in a single variable to be controlled, $y(t)$, and a single control variable, $x(t)$, where t is time. A trained ANN can be used to predict $y^{i+1} = y((i+1)\Delta t)$ knowing $y^i = y(i\Delta t)$ if the system satisfies an unknown differential equation of first order. If the equation is of higher order, one can, in principle, include sufficient time-derivatives of a vector version of $y(t)$ to reduce all scalar equations in its components to first order. In practice this is difficult since analog signals corresponding to derivatives may not be readily available. However, a finite difference approximation of the derivatives shows that, provided Δt is small enough, this is equivalent to adding information about y at previous instants in time, i.e., y^{i-1}, y^{i-2}, etc., as many as needed. Previous values of $x^i = x(t)$ may also be needed for systems of higher order. A memory unit, which is part of the computerized control system, stores y^i and x^i to provide information on previous instants to the ANN.

A regular input−output model is shown in Figure 1(a). It shows an open-loop control system for which the control variable x^i is known. This is the



(a)

(b)

FIGURE 1. Neurocontrollers: (a) open loop and (b) closed loop. For clarity, the plant to be controlled is not shown.

variable that is manipulated to obtain the desired behavior from the plant to be controlled. For clarity, only the controller is shown in the figure, but not the plant. The number of inputs and outputs of the ANN are not the same. Though this is not a problem for a neural network, it is awkward for stability analysis. For this purpose, a vector $\mathbf{u}$ of dimension n will be defined such that on the input side of the ANN

$$\mathbf{u}^i = (y^{i-n+1}, y^{i-n+2}, \ldots, y^{i-1}, y^i), \tag{1}$$

and on the output side

$$\mathbf{u}^{i+1} = (y^{i-n+2}, y^{i-n+3}, \ldots, y^i, y^{i+1}). \tag{2}$$

The overall effect of the control system is that of a nonlinear map

$$\mathbf{u}^{i+1} = \mathbf{F}(\mathbf{u}^i). \tag{3}$$

The variables corresponding to these that will be actually used are defined in "Experimental Verification."

For closed-loop operation, a controller generates the control variable $x(t)$, as shown in Figure 1(b). The controller is modeled by

$$x^i = g(y^i, y^{i-1}, \ldots, x^{i-1}, x^{i-2}, \ldots, y^*), \tag{4}$$

and the ANN by

$$y^{i+1} = f(y^i, y^{i-1}, \ldots, x^i, x^{i-1}, x^{i-2}, \ldots), \tag{5}$$

where y^* is a reference value of the controlled variable. Once again the map represented by Eq. (3) applies where

$$u^i = (y^{i-n+1}, y^{i-n+2}, \ldots, y^{i-1}, y^i, x^{i-m+1}, x^{i-m+2}, \ldots, x^{i-1}, x^i). \tag{6}$$

Consider the nonlinear map (3) for open-loop control, or a similar map obtained by using Eqs. (4) and (5) in closed-loop control. The fixed points $\bar{\mathbf{u}}$ map to themselves and are hence solutions of $\bar{\mathbf{u}} = \mathbf{F}(\bar{\mathbf{u}})$. A nonlinear map can have more than one fixed point with different stability characteristics. To study the stability of a given fixed point, the map is linearized around it to get

$$\mathbf{u}^{i+1} - \bar{\mathbf{u}} = \mathbf{J}(\mathbf{u}^i - \bar{\mathbf{u}}), \tag{7}$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{F}$ evaluated at the fixed point. The spectral radius of $\mathbf{J}$, denoted by r, is the largest of the absolute values of its eigenvalues. If $r < 1$, the images of the map converge to the fixed point and it is stable; otherwise, it is unstable (Hale and Kocak 1991).

STABILITY ANALYSIS

In this section, the behavior of neural networks is analyzed, going progressively from simple to more complex configurations. The simple ones

are addressed analytically, while the more complex are numerically computed. In the latter case, the Jacobians were calculated using second-order accurate numerical derivatives, and the spectral radii were found using the implicit double-shifted QR algorithm of the EVCRG routine in the IMSL[1] library. For validation, the results of the numerical code for the 2-1-1 network were compared with the analytical solution, and the spectral radii from both methods were found to be identical.

Open-Loop Control with Single Neuron

As ANNs are composed of a system of neurons connected together to build a network, the stability of one of these is first analyzed. Consider two inputs to a single neuron, y^i and x, where x is a control parameter that is kept constant, and an output, y^{i+1}. There are two synaptic weights, w_{11}^{21} and w_{12}^{21}, and only one bias, θ_{21}. Defining $\mathbf{u}^i = (y^i, y^{i+1})$, we get the map

$$u_1^{i+1} = u_2^i, \tag{8}$$

$$u_2^{i+1} = f(u_2^i), \tag{9}$$

where

$$f(u_2^i) = [1 + \exp\{-\theta_{21} - w_{11}^{21} u_2^i - w_{12}^{21} x\}]^{-1}. \tag{10}$$

The fixed point of this map is $(\bar{u}_1 = \bar{u}_2)$ where $\bar{u}_1 = \bar{u}_2$ and $\bar{u}_2 = f(\bar{u}_2)$. Depending on the weights, there may be more than one fixed point. The Jacobian at the fixed point is

$$\mathbf{J} = \begin{bmatrix} 0 & 1 \\ 0 & \lambda_2 \end{bmatrix}, \tag{11}$$

where $\lambda_k = \overline{\partial f / \partial u_k^i}$; the overbar indicates that the derivative is evaluated at the fixed point. In this simple case, λ_2 is the slope of the curve represented by Eq. (9). We have $\lambda_2 = w_{11}^{21} e^{-\beta}(1 + e^{-\beta})^{-2}$, where $\beta = \theta_{21} + w_{11}^{21}\bar{u}_1 + w_{12}^{21} x$. The eigenvalues of $\mathbf{J}$ are 0 and λ_2 so that the spectral radius is $r = |\lambda_2|$.

The fixed point is stable if $r < 1$. The key to stability is Eq. (9), which can be linearized to $u_2^{i+1} - \bar{u}_2 = \lambda_2(u_2^i - \bar{u}_2)$. Thus, for $0 < \lambda_2 < 1$, each u_2 will be successively close to $\bar{u}_2$ and on the same side of it. If, however, $-1 < \lambda_2 < 0$, then u_2 will oscillate around $\bar{u}_2$ as it approaches. The sign of λ_2 is determined by the sign of w_{11}^{21}. There are two possible dynamic behaviors that can be found if the fixed point is unstable: The system may oscillate around the fixed point, or it may drift to another fixed point that is stable but undesirable.

Figures 2, 3, and 4 show different behaviors depending on w_{11}^{21}. w_{12}^{21} and θ_{21} have been kept constant. In each figure, the upper graph shows y_2^{i+1} vs. i_1, the center one y_2^{i+1} vs. y_2^i, and the lowest r vs. i. In Figures 2

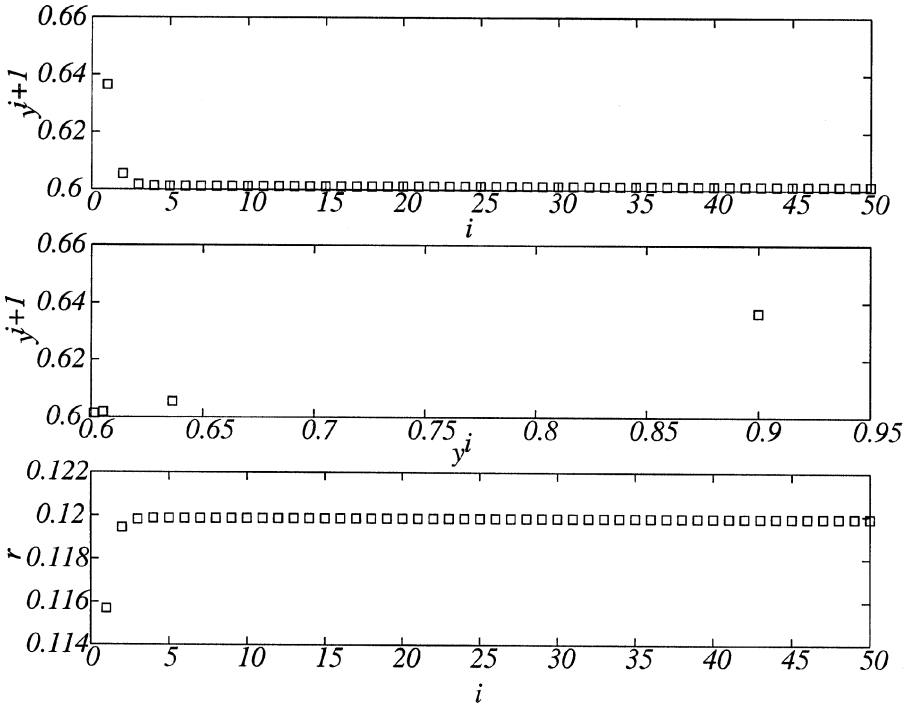


FIGURE 2. Single neuron, stable behavior without oscillations, $\theta_{21} = 0.1, w_{12}^{21} = 0.1, w_{11}^{21} = 0.5$.

and 3, the spectral radius is less than unity and the fixed point is stable. In Figure 2, w_{12}^{21} is positive and less than unity so that there is monotonic convergence, while it is negative in Figure 3, leading to oscillations. For Figure 4, $r > 1$ so that the fixed point is not stable. Since $w_{12}^{21} < 0$, an initial condition near it wanders away in an oscillatory fashion, ending eventually in constant amplitude oscillations. Thus, the stability of a map of the type of Eqs. (8) and (9) and the nature of its behavior around a fixed point depend on the value and sign of λ_2, which is a function of the synaptic weights and biases.

Open-Loop Control with 2-1-1 Neural Network

The purpose of this example is to show that even though the steady-state prediction of two ANNs trained with the same data but with different initial weights and biases might be identical, their dynamic response can differ significantly. Training of a simple ANN with an equally simple analytical function allows us to explain the concepts without excessive and unnecessary algebra. One of the simplest ANN is that with two input neurons, one hidden layer with one neuron, and a single output neuron, i.e., a 2-1-1 structure. The simplest set of data, a single point on $y = x^2$ with $x = 0.7$, was provided to the ANN. It was trained using a gradient method (Pierre 1986), which

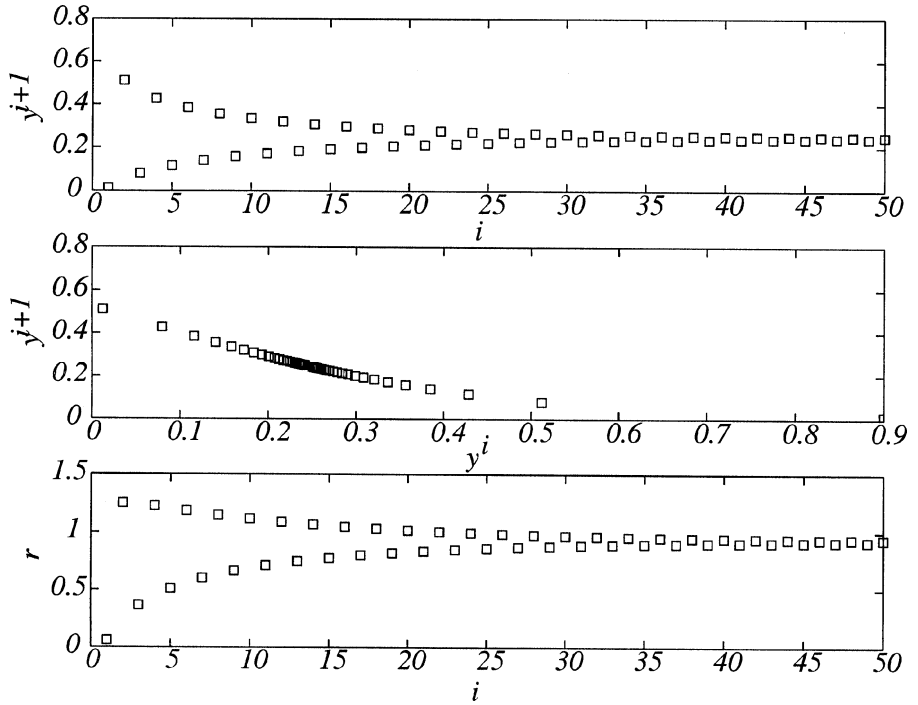


FIGURE 3. Single neuron, stable behavior with oscillations, $\theta_{21} = 0.1, w_{12}^{21} = 0.1, w_{11}^{21} = -5.0$.

minimizes the error $e = \frac{1}{2}(y^* - y)^2$, where y^* is the target value and y is the prediction of the ANN.

Writing $\mathbf{u}^i = (y^i, y^{i+1})$ again, the map is represented by Eqs. (8) and (9), where

$$f(u_2^i) = \left[1 + \exp\left\{-\theta_{31} - \frac{w_{21}^{31}}{1 + e^{-\beta}}\right\}\right]^{-1} \tag{12}$$

and $\beta = \theta_{21} + w_{11}^{21} u_1^i + w_{12}^{21} x$. The spectral radius is $r = |\lambda_2|$, where $\lambda_2 = w_{21}^{31} w_{11}^{21} e^{-\beta} e^{-\alpha} (1 + e^{-\alpha})^{-2} (1 + e^{-\beta})^{-2}$ and $\alpha = \theta_{31} + w_{21}^{31}/(1 + e^{-\beta})$.

The stability of the map depends on the trained weights and biases, which in turn depend on the initial values used during training. In Table 1, runs A and B show the initial and final weights and biases of the network and the final spectral radii of the Jacobian of the maps that result from training with two different sets of initial weights and biases. Though the error in each case is zero, it can be seen that different initial weights and biases may lead to different values after training.

Figure 5 shows the time-dependent behavior of the network for case A. The system is stable at the prescribed fixed point and $y^i \to \bar{y}$ as $i \to \infty$. Once

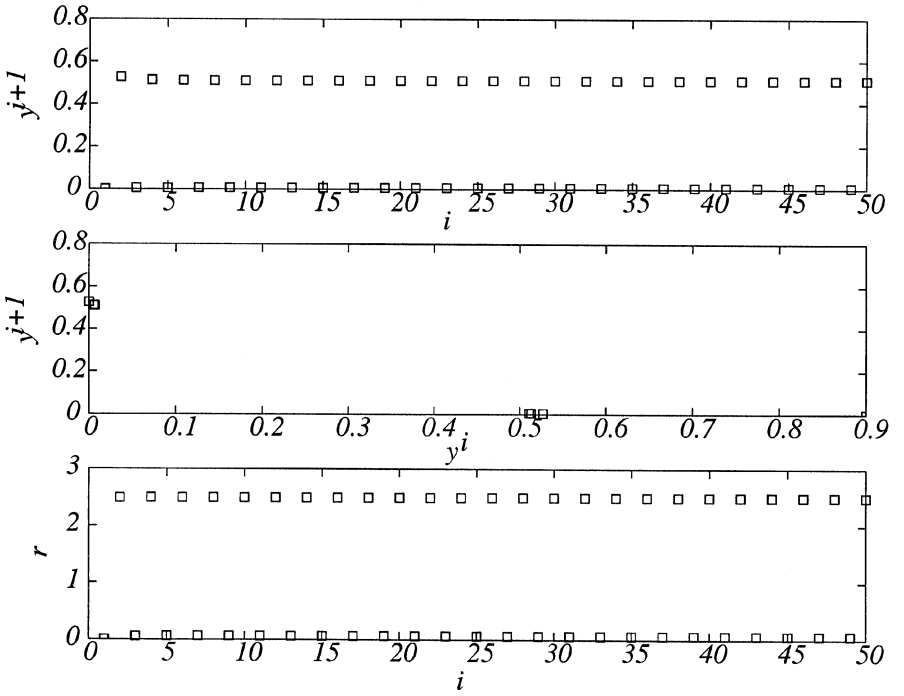


FIGURE 4. Single neuron, unstable behavior, $\theta_{21} = 0.1, w_{12}^{21} = 0.1, w_{11}^{21} = -10.0$.

the fixed point is predicted correctly, the spectral radius takes the value obtained in the training process. On the other hand, Figure 6 shows the output of the same ANN configuration but now using the weights for case B. The fixed point is not stable, and the long-time behavior of the map is seen to be oscillatory around it with constant amplitude. Though both A and B are able to make good steady-state predictions, one control system is stable and the other is not.

TABLE 1 2-1-1 Neural Network, Initial, and Final Weights and Biases and Final Spectral Radii. Cases A and B are without and C is with stabilization.

		w_{11}^{21}	w_{12}^{21}	w_{21}^{31}	θ_{21}	θ_{31}	r
A	Initial	0.5	− 0.7	0.3	− 0.1	− 0.1	—
	Final	0.4981	− 0.7026	0.2774	− 0.1038	− 0.1545	0.0084
B	Initial	9.0	− 9.0	− 9.0	− 0.1	− 0.1	—
	Final	8.781	− 9.311	− 8.945	− 0.5450	0.4925	1.0991
C	Initial	9.0	− 9.0	− 9.0	− 0.1	− 0.1	—
	Final	1.261	− 6.281	− 8.936	3.783	8.862	0.1068

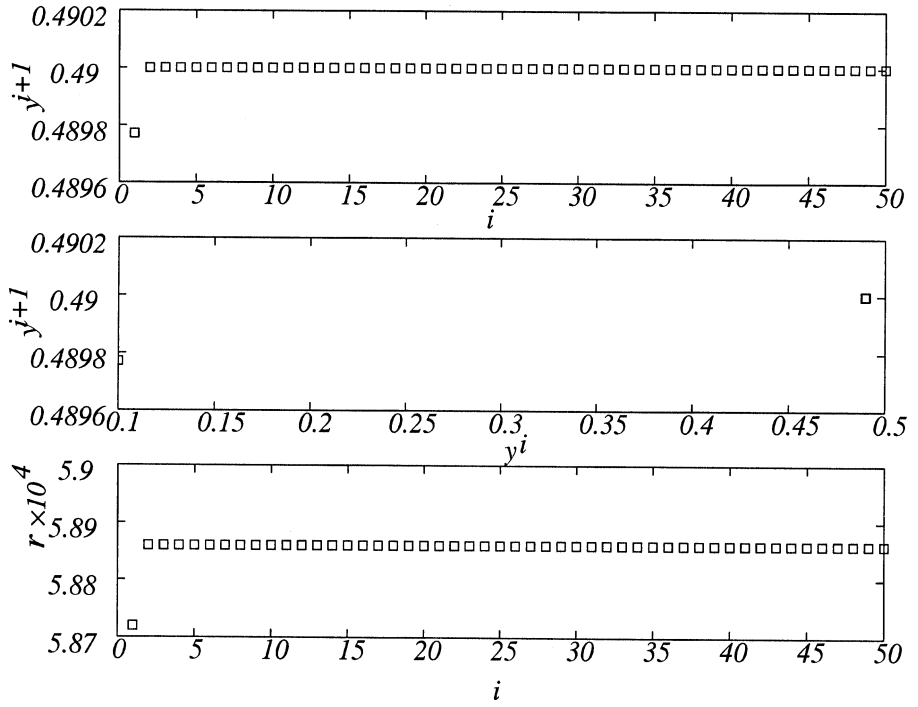


FIGURE 5. 2-1-1 network, case A.

Open-Loop Control with 2-5-1 Neural Network

The purpose of this section is to show the relation between the dynamic behavior of an ANN and the value of the spectral radius of the linearized iterative map. This is trained using a gradient method by providing only one set of data to the network, i.e., $y = x^2$ with $x = 0.7$. The inputs to the ANN are y and x and the output is y. The network is trained starting from two different initial weights and biases. In both cases, call them D and E, the error is reduced by training to zero, but the spectral radii converge to 0.0915 and 1.069, respectively. The ANN is now used as an open-loop controller as in Figure 1(a). Figure 7 shows $y^{i+1} = f(y^i)$ for both sets of weights where the fixed points are determined by the intersection of the curve with the $y^{i+1} = y^i$ line. For case D, there is only one fixed point. It is stable since the absolute value of its slope at the fixed point is less than unity as indicated by its intersection with the $y^{i+1} = y^i$ line. On the other hand, for case E, there are three fixed points: two stable and one unstable. The network is trained to the unstable fixed point and the behavior of the open-loop control system is indicated in Figure 8. The system goes to the unstable fixed point, $y^{i+1} = 0.49$, and then moves away to a stable one, $y^{i+1} = 0.72$. The spectral radius at the

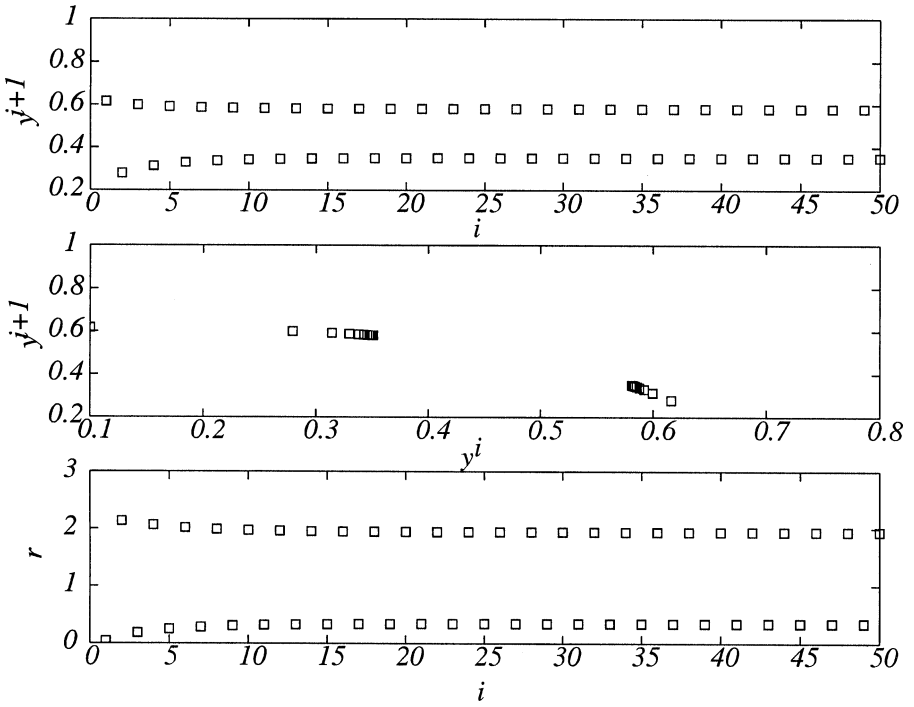


FIGURE 6. 2-1-1 network, case B.

second fixed point is $r = 0.847$. Though it is stable, the prediction of the ANN is incorrect since the fixed point is not the one that it was trained for.

TRAINING WITH STABILIZATION

It is seen that training may produce an ANN with good steady-state prediction but which is unstable in dynamic operation as a control system. To avoid this, we propose a training method to drive the ANN to weights and biases that also guarantee the stability of the desired fixed point. In this method, the usual error minimization is made to alternate with a stabilization procedure.

The stabilization is switched on if the system is in danger of becoming unstable, i.e., when it is found that the spectral radius r is greater than unity during the error minimization process. The objective of the stabilization is to reduce r, which can be done using a simple gradient procedure. The effect of each one of the weights w_{ij}^{kl} on r is determined by the derivative $\partial r/\partial w_{ij}^{kl}$. These are calculated and the weights are then changed according to

$$\Delta w_{ij}^{kl} = -\eta \frac{\partial r}{\partial w_{ij}^{kl}}, \tag{13}$$

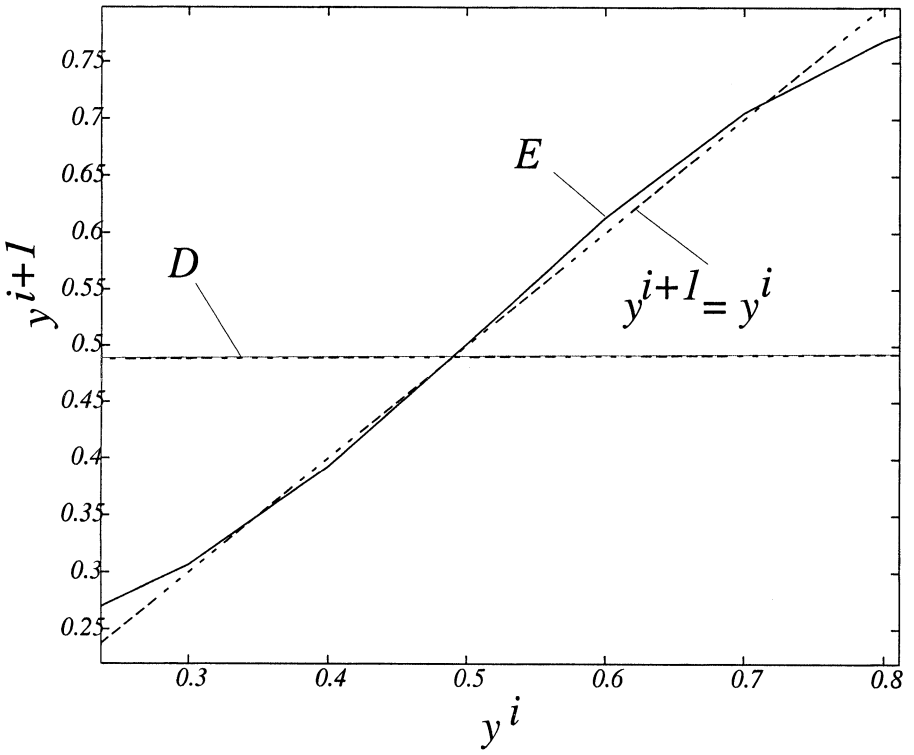


FIGURE 7. 2-5-1 network, y^{i+1} vs. y^i for stable (D) and unstable (E) maps; $y^{i+1} = y^i$ line is shown for reference.

where Δw_{ij}^{kl} is the correction to the weight w_{ij}^{kl}, and η is a relaxation parameter (taken here to be 0.1). This is a first-order gradient formula for reduction of r. A tolerance criterion, $r < 0.5$ say, can be set with which we are comfortable about the stability of the controller. Once this condition is met, we can resume minimization of the error.

This procedure is tested with the simple 2-1-1 network discussed before with open-loop control. The errors during training can be calculated as functions of the three weights, w_{11}^{21}, w_{12}^{21}, and w_{21}^{31}, and two biases, θ_{21} and θ_{31}. To show in graphical form, one weight and two of the biases are fixed so that the error, e, and spectral radius, r, can be shown as functions of weights w_{11}^{21} and w_{21}^{31}. Figures 9 and 10 show the contours of $e(w_{11}^{21}, w_{21}^{31})$ and $r(w_{11}^{21}, w_{21}^{31})$. The thick line in both figures corresponds to $e = 0$, which is the goal of the training. There are infinitely many rs, some greater than and some smaller than unity, that have $e = 0$. A stable, trained network can be obtained by moving towards the $e = 0$ line, alternating with reduction in r. We begin with initial weights, corresponding to point a in both figures. The lines abc indicate the path that would be taken by a gradient error minimization procedure alone.

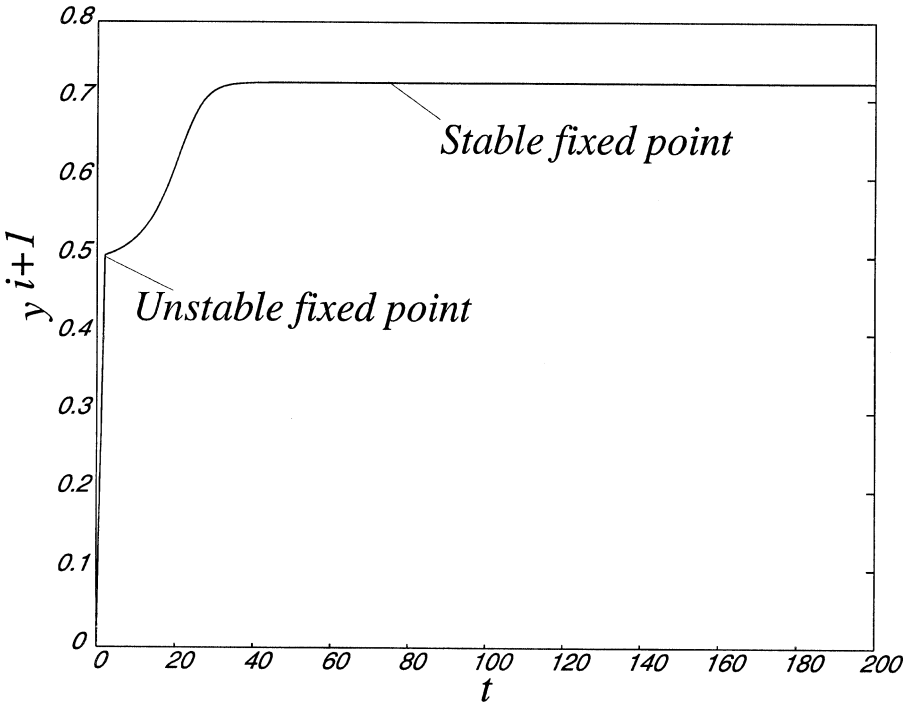


FIGURE 8. 2-5-1 network, unstable open-loop control system.

At the end, we would have $e = 0$ and $r = 1.39$. Compare this with the inclusion of the stabilization procedure which, as an example, is made to turn on whenever $r < 0.5$. The path from a to b, where $r = 0.5$, is the same as before. However, after that, the weights and biases are taken from b to d through a zigzag path around $r = 0.5$. The final values at d are $e = 0$ and $r = 0.43$, indicating a stable open-loop control system.

The results of training the ANN, using the same initial values as in case B, but with stabilization, is shown as row C in Table 1. With this set of weights and biases, the open-loop control system is now stable and the desired fixed point is reached. Figure 11 shows the behavior of the system that has been stabilized and can be compared to Figure 6, which was not.

EXPERIMENTAL VERIFICATION

The experiments were carried out in the Hydronics Laboratory of the Department of Aerospace and Mechanical Engineering of the University of Notre Dame. Several years ago, a specially designed heat-exchanger test facility, shown in Figure 12, was built for thermal control experiments (Zhao 1995). This facility consists of an in-draft wind tunnel facility in which

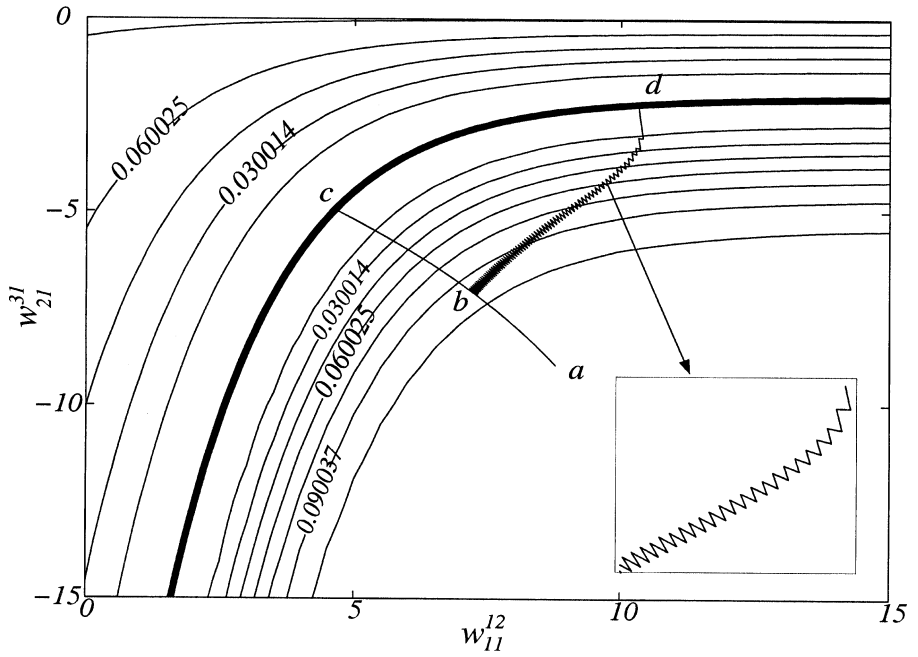


FIGURE 9. 2-1-1 network error contours; thick line is $e = 0$; *abc* error minimization alone; *abd* error minimization with stabilization; inset shows zig-zag path.

water-to-air heat exchangers can be placed. The motion of the air is due to a fan that is controlled by a variable speed drive; for safety, there is an upper bound to the allowable air speed, U_{max}. A PID-controlled electrical resistance heater provides hot water. Thermocouples are used to sense the air temperature downstream of the heat exchanger, T_{out}^{air}. A Pitot tube located upstream of the heat exchanger is connected to a differential pressure transducer to give the air flow rate, U. The data are fed to a PC running on LabVIEW, which are then processed by a controller which manipulates the air-flow rate. For the present experiments, a nominal, 18 in. × 24 in. water coil water-to-air fin-tube compact heat exchanger[2] was used.

Previous publications have used this facility in a series of experiments for neural network simulation and control purposes. The steady-state behavior of heat exchangers was modeled by Díaz et al. (1999). In these experiments, the system was allowed to reach thermal equilibrium before measurements were made, and the objective was to obtain input-output pairs for training of an ANN. Different network configurations were studied in order to select the best. The unsteady behavior of heat exchangers was also modeled using ANNs and described by Díaz et al. (2001a; 2001b). Once the dynamics can be reliably predicted, internal model control methods can be used to manipulate the flow rates to obtain a desired set point temperature of the outlet air. The

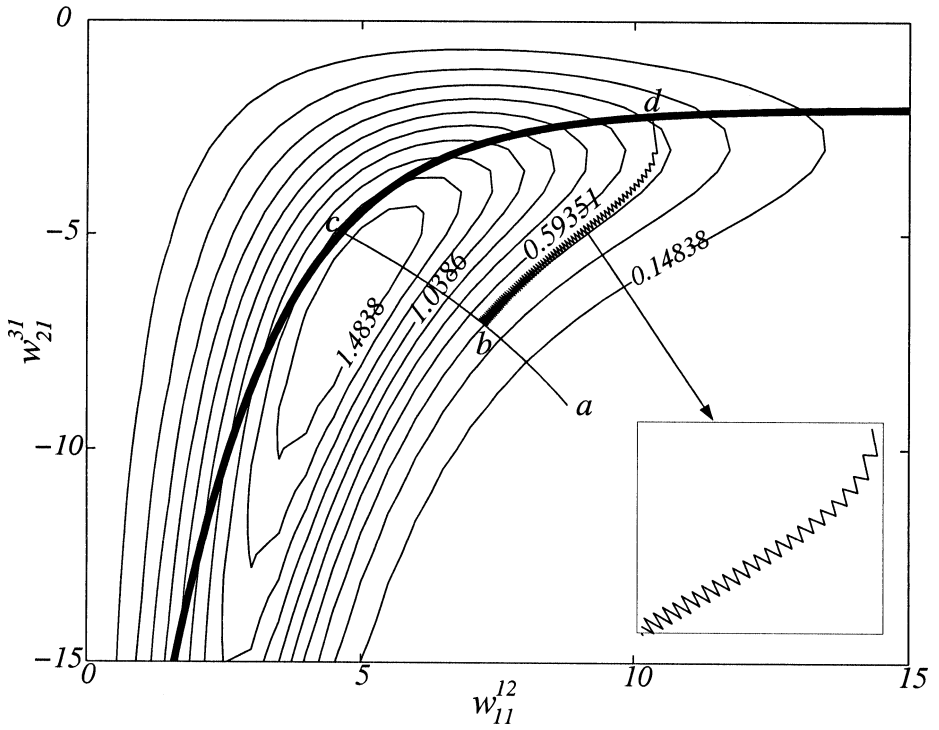


FIGURE 10. 2-1-1 network spectral radius contours; thick line is $e = 0$; *abc* error minimization alone; *abd* error minimization with stabilization; inset shows zig-zag path.

performance of the controller was compared to standard techniques such as PID.

In a parallel publication (Díaz et al. 2002), the details of the stabilization techniques that have been developed here to regulate the temperature of the air coming out of the heat exchanger have been described. On the basis of previous experience, a multilayer ANN was preferred even though the methodology is perfectly general; specifically, the ANNs used in the control system had a 6-10-5-1 configuration. The inputs to the network correspond to T_{out}^{air} and voltage to the variable speed drive. Three values of these two variables taken at consecutive instants in time were used. Thus, (y^{i-2}, y^{i-1}, y^i) correspond to values of T_{out}^{air} and (x^{i-2}, x^{i-1}, x^i) to values of the voltage to the variable speed drive. The variables were sampled by the computer at a time interval of Δt around one second; this is fast enough compared to the time rates of change in the heat exchanger.

The difference between stable and unstable behaviors of the controller was demonstrated using two different controllers with $r < 1$ and $r > 1$, respectively. The weights for the stabilized controller were found using the algorithm proposed here, Eq. (13), while the unstable controller was found

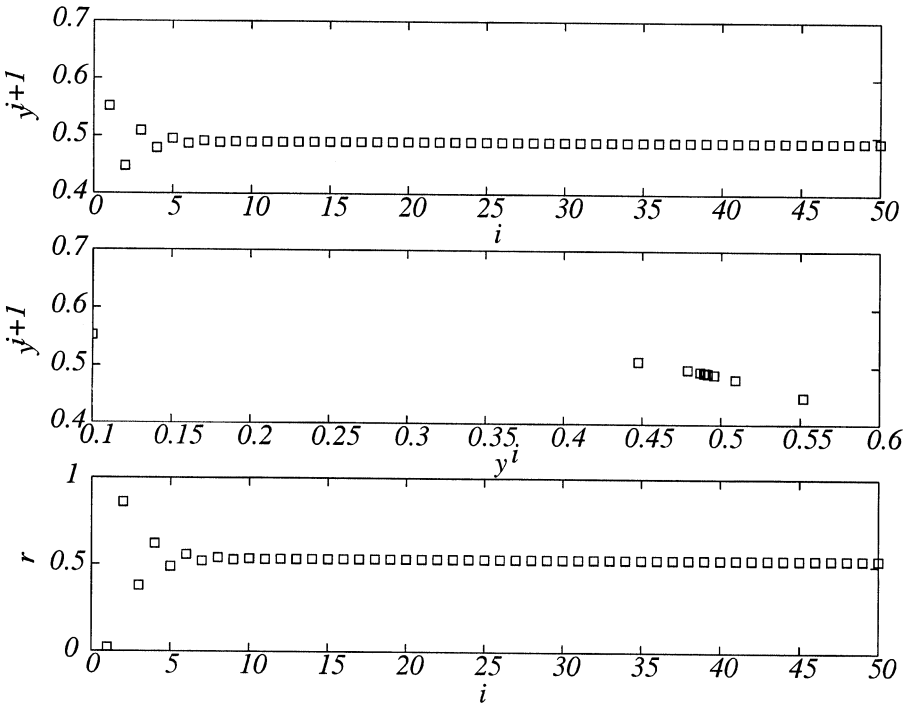
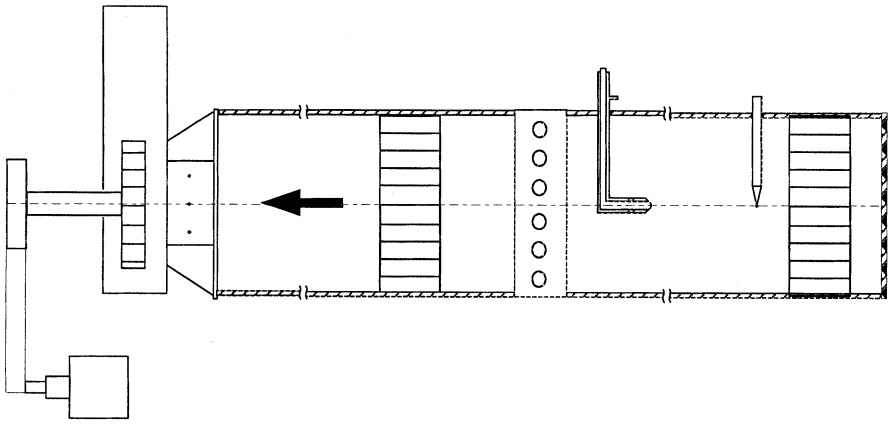


FIGURE 11. 2-1-1 network, case C.

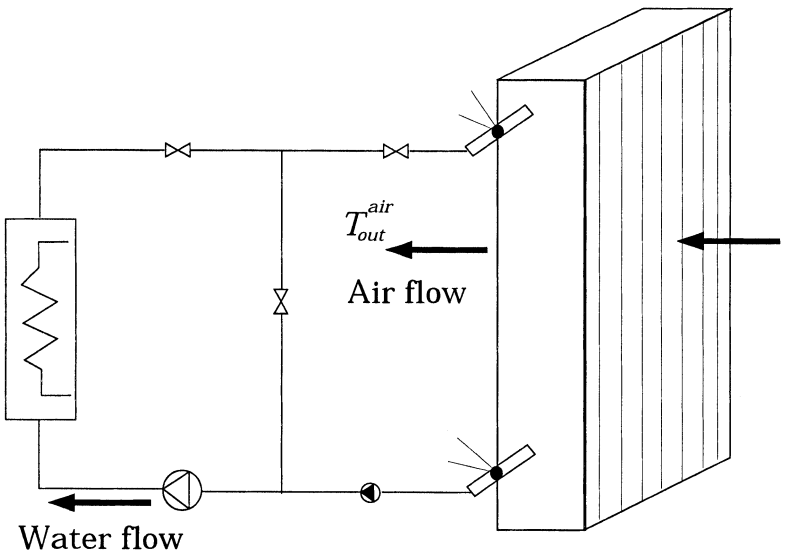
by using the same equation but with a change in sign in the right side to drive the weights in a direction that makes the controller unstable. The reference temperature was chosen to be $T_{ref} = 34.0°C$. The flow rates and temperatures in the test facility were first adjusted to get T_{out}^{air} as close to T_{ref} as possible, and then the controller was turned on. Figure 13 shows the response of the two controllers. For the stabilized controller, T_{out}^{air} oscillates a few times but eventually goes to T_{ref}. On the other hand, for otherwise identical conditions, the unstable controller takes the system to the maximum air speed, U_{max}, where it remains without accomplishing the control task.

CONCLUSIONS

Controllers used for thermal systems should be stable. It is thus important that, if neural networks are used for this purpose, the training procedures be such that the resulting system is guaranteed to be stable. This is especially true if the network is adaptive (Díaz et al. 2001b), such as in thermal systems that evolve over time either due to fouling or changes in hardware.



(a)



(b)

FIGURE 12. Experimental setup: (a) heat exchanger test facility with wind tunnel and in-draft fan, (b) heat exchanger with water air flows indicated; T_{out}^{air} is the air outlet temperature.

The stability of a thermal open- or closed-loop neurocontroller is determined by the spectral radius of the Jacobian of the map that governs the process. This is easily calculated for small networks. A training algorithm for networks is proposed here that can find a set of weights and biases that reduces the target error to a minimum, but for which the control system is

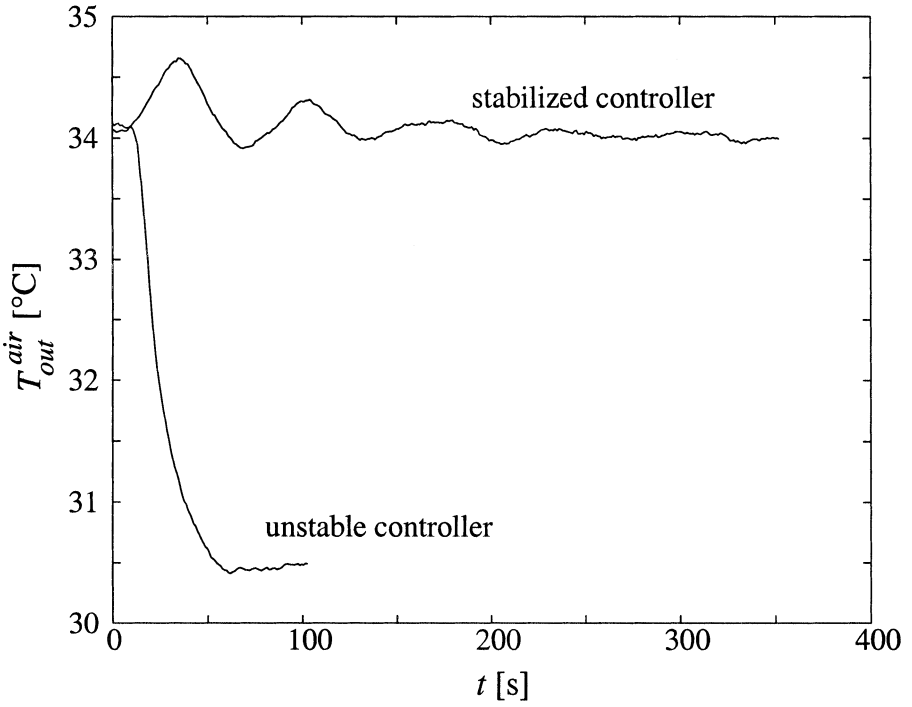


FIGURE 13. Experimental operation of stabilized and unstable controllers for control of T_{air}^{out}.

also stable. This technique was experimentally tested and verified on the control of the air temperature coming out of a water-air heat exchanger.

NOMENCLATURE

e	error
$\mathbf{F}$	mapping function
f	function for model of plant
g	function for controller
(i,j)	neuron j in layer i
$\mathbf{J}$	Jacobian matrix of linearized map
m	number of components for control variable
n	number of components for controlled variable
r	spectral radius of Jacobian matrix
T_{ref}	reference temperature [°C]
T_{out}^{air}	outlet air temperature [°C]
t	time
Δt	time interval
U	air velocity

U_{max} maximum air velocity
$\mathbf{u}$ mapping vector
$w_{i,j}^{k,l}$ synaptic weight between neurons (i,j) and (k,l)
$\Delta w_{i,j}^{k,l}$ correction to weight $w_{i,j}^{k,l}$
x control variable
y controlled variable
y^* reference value of controlled variable

Greek Symbols

η relaxation parameter
$\theta_{i,j}$ bias of neuron (i,j)
λ_k $= \overline{\partial f / u_k^i}$

Subscripts and Superscripts

$\underline{i}$ time index
$\overline{()}$ value at fixed point

NOTES

1. Bristol Technology, Inc., Version 3.1.
2. Type T, manufactured by Trane, Inc., La Crosse, WI 54601.

REFERENCES

AlDuwaish, H. N., and M. N. Karim. 1996. New methodology for identification and control of plants with static input or output nonlinearities. *Computers and Chemical Engineering*, Suppl. B 20:S993–S998.

Bittanti, S., and L. Piroddi. 1997. Nonlinear identification and control of a heat exchanger: A neural network approach. *J. Franklin Inst.*, 334B (1):135–153.

Dasgupta, M. S., C. B. Menon, and R. K. Gupta. 2001. ANN controller trained with steady state input-output data for a heat exchanger. *Indian Journal of Chemical Technology*, 8(3):227–234.

Delgado, A. 1998. Stability analysis of neurocontrol systems using a describing function. In *Proceedings of 1998 IEEE International Joint Conference on Neural Networks*, IEEE World Congress on Computational Intelligence 3, May 4–9, Anchorage, AK, USA. pages 2126–2130.

Díaz, G., M. Sen, K. T. Yang, and R. L. McClain. 1999. Simulation of heat exchanger performance by artificial neural networks. *HVAC&R Research Journal*, 5(3):195–208.

Díaz, G., M. Sen, K. T. Yang, and R. L. McClain. 2001a. Dynamic prediction and control of heat exchangers using artificial neural networks. *International Journal of Heat and Mass Transfer* 44(9): 1671–1679.

Díaz, G., M. Sen, K. T. Yang, and R. L. McClain. 2001b. Adaptive neurocontrol of heat exchangers. *ASME Journal of Heat Transfer* 123(3):417–612.

Díaz, G., M. Sen, K. T. Yang, and R. L. McClain. 2002. Stabilization of a neural network-based temperature controller for heat exchangers. In *Proceedings of the 12th International Heat Transfer Conference*, pages 225–230, Volume 4, Grenoble, France.

Hale, J., and H. Kocak. 1991. *Dynamics and Bifurcations*. New York: Springer-Verlag.

Hrycej, T. 1995. Stability and equilibrium points in neurocontrol. In *Proceedings of 1995 IEEE International Conference on Neural Networks*, 1, pages 617–621.

Jin, L., M. Gupta, and P. Nikiforuk. 1993. Stable dynamic backpropagation using constrained learning rate algorithm. In *Proceedings of 1993 International Joint Conference on Neural Networks*, 1, pages 2654–2657.

Matko, D., K. Kavsek-Biasizzo, and J. Kocijan. 1998. Neuro-fuzzy model-based control. *Journal of Intelligent and Robotic Systems* 23(2–4):249–265.

Pacheco-Vega, A., M. Sen, K. T. Yang, and R. L. McClain. 2001a. Neural network analysis of fin-tube refrigerating heat exchanger with limited experimental data. *International Journal of Heat and Mass Transfer* 44:763–770.

Pacheco-Vega, A., G. Díaz, M. Sen, K. T. Yang, and R. L. McClain. 2001b. Heat rate predictions in humid air-water heat exchangers using correlations and neural networks. *ASME Journal of Heat Transfer* 123(2):348–354.

Pierre, D. A. 1986. *Optimization Theory with Applications*. New York: Dover Publications, Inc.

Pineda, F. 1987. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters* 59(19):2229–2232.

Quek, C., and A. Wahab. 2000. Real-time integrated process supervision. *Engineering Applications of Artificial Intelligence* 13(6):645–658.

Ramasamy, S., P. B. Deshpande, S. S. Tambe, and B. D. Kulkarni. 1995. Robust non-linear control with neural networks. In *Proceedings of the Royal Society of London Series A-Mathematical And Physical Sciences* 449(1937), pages 655–667.

Riverol, C., and V. Napolitan. 2000. Use of neural networks as a tuning method for an adaptive PID: Application in a heat exchanger. *Chemical Engineering Research and Design* 78(A8):1115–1119.

Rumelhart, D. E. (ed.) and J. L. McClelland. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, pages 318–362. Cambridge, MA: The MIT Press.

Sen, M., and K. T. Yang. 2000. Applications of artificial neural networks and genetic algorithms in thermal engineering. In *The CRC Handbook of Thermal Engineering*, Section 4.24 ed. F. Kreith, 620–661 CRC Press, Boca Raton, FL, USA.

Zhao, X. 1995. *Performance of a Single-Row Heat Exchanger at Low In-Tube Flow Rates*, M.S. Thesis, Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN 46556.

Copyright of Applied Artificial Intelligence is the property of Taylor & Francis Ltd and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.