Switching Controllers Based on Neural Network Estimates of Stability Regions and Controller Performance

Enrique D. Ferreira and Bruce H. Krogh

Department of Electrical and Computer Engineering
Carnegie Mellon University

email: edf/krogh@ece.cmu.edu

Abstract. This paper presents new results on switching control using neural networks. Given a set of candidate controllers, a pair of neural networks is trained to identify the stability region and estimate the closed-loop performance for each controller. The neural network outputs are used in the on-line switching rule to select the controller output to be applied to the system during each control period. The paper presents architectures and training procedures for the neural networks and sufficient conditions for stability of the closed-loop system using the proposed switching strategy. The neural-network-based switching strategy is applied to generate the switching strategy embeded in the SIMPLEX architecture, a real-time infrastructure for soft on-line control system upgrades. Results are shown for the real-time level control of a submerged vessel.

1 Introduction

A common approach to control complex dynamic systems is to design a number of different controllers, each for a particular operating region or performance objective, and then to switch among the controllers in real time to achieve the overall control objective. This is, for example, the philosophy behind gain-scheduled controllers. Recently, switching control strategies have been proposed for adaptive control of unknown systems [1],[9], and to optimize the performance of stabilizing controllers for a known plant [8].

It is useful to view switching control systems as hybrid systems, that is, systems with both continuous state variables and discrete state variables. The plant state variables (assuming a continuous-variable system) and possibly continuous state variables in the controllers constitute the continuous state of the switching control system; the index of the current controller being applied to the system, and possibly discrete variables in the sequential switching logic, constitute the discrete state. System performance and stability are also normally defined in terms of the continuous-state trajectories. Analysis of switching control systems from

either perspective is difficult because of the interaction between the continuous and discrete dynamics through the switching rules.

Given a collection of controllers for a nonlinear dynamic system, neural network techniques are presented for estimating the regions of stability and performance of each controller, and an on-line switching strategy is proposed based on these neural network estimates. Sufficient conditions are presented for closed-loop stability of the switching control system. On the application side, we describe the use of the neural network strategy to implement the switching rules in the SIMPLEX architecture, a real-time environment developed at the Software Engineering Institute at Carnegie Mellon University that provides protection against errors in control system upgrades [11],[4] and [10]. Results are presented for the real-time control of the level of a submerged vessel.

2 Problem Formulation

We consider the problem of controlling a nonlinear system described by the state equations

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{x}_k \in S, &\qquad \mathbf{u}_k \in D \end{aligned} \tag{1}$$

where $\mathbf{x}_k \in R^n$ is the state vector and $\mathbf{u}_k \in R^m$ the control input vector. The connected sets $S \subset R^n, D \subset R^m$ represent physical constraints on the system state and control, respectively. The discrete-time state equation reflects the sampled-data implementation of a computer control system. The control objective is to take the state to the origin.

We assume M state feedback controllers have been designed for this system, with the i^{th} control law given by $\mathbf{g}^i : R^n \rightarrow R^m$. We assume the origin is a stable equilibrium for each of the controllers in some (unknown) region. The objective of the switching strategy is to select one of the control outputs to apply the system at each control instant to achieve the largest possible region of stability for the closed-loop system with a good transient response.

The closed-loop system created by the application of each controller is characterized by a stability region and a performance index. The region of stability for controller i is defined as

$$\begin{aligned} R^i = \{\ \mathbf{x}_o : \mathbf{x}_k^i(\mathbf{x}_o) \in S,\ \mathbf{g}^i(\mathbf{x}_k^i(\mathbf{x}_o)) \in D \quad \forall k \geq k_o \quad \text{and} \\ \lim_{k \rightarrow \infty} \mathbf{x}_k^i(\mathbf{x}_o) = 0\ \} \end{aligned} \tag{2}$$

where $\mathbf{x}_k^i(\mathbf{x}_o)$ denotes the trajectory of the system (1) with the initial state $\mathbf{x}_o$ at $k = 0$ under control law $\mathbf{g}^i$. We consider performance indices for the controllers of the form

$$J_\delta^i(\mathbf{x}_o) = B^i + \sum_{k=0}^{\infty} \delta^k U^i(\mathbf{x}_k^i(\mathbf{x}_o)), \quad i = 1, \ldots, M \tag{3}$$

where $0 < \delta \leq 1$ is a discount factor, $U^i : R^n \rightarrow R$ is a positive definite *state cost function*, and B^i is the *bias coefficient* for the i^{th} controller. We assume (3) converges for $\delta = 1$.

The proposed approach for selecting the controller at each sampling instant is illustrated in figure 1. Neural networks are used to compute estimates of the stability regions R^i and performance indices J^i_δ at the current state, denoted by $\hat{R}^i$ and $\hat{J}^i_\delta$, respectively. The index of the control input to be applied for the next period, denoted i_k, is then selected as

$$i_k = arg \min_{i \in I(\mathbf{x}_k, L_k)} \{\hat{J}^i_\delta(\mathbf{x}_k)\} \tag{4}$$

where

$$I(\mathbf{x}_k, L_k) = \{i | \mathbf{x}_k \in \hat{R}^i, \text{and if } i \neq i_{k-1}, \hat{J}^i_\delta(\mathbf{x}_k) < L^i_k\}$$

with $L_k = \{L^1_k, \ldots, L^M_k\}$ and for $i = 1, \ldots, M$

$$L^i_k = \begin{cases} \infty & \text{if } k = 0 \text{ or } \mathbf{x}_k \notin \hat{R}^{i_{k-1}} \\ L^i_{k-1} & \text{if } i \neq i_k \\ \min\{\hat{J}^i_\delta(x_k), L^i_{k-1}\} & \text{if } i = i_k \end{cases}$$

In words, the scheduler selects the controller with the minimum estimate performance index from among the controllers for which the current state is in the estimated stability region and, for the controllers other than the current controller, the current estimated performance index is less than the corresponding bound L^i_k. The limits L^i_k guarantee a controller is not re-selected once it has been used until its performance index has decreased below the lowest value reached when it was last active. If the system leaves the stability region of the controller used during the previous period, all controllers become candidates again by setting $L^i_k = \infty$ for all $i = 1, \ldots, M$. This selection criterion is motivated by the *min-switching* strategies based on Lyapunov functions proposed in [8] and [6]. The strategy proposed in this paper replaces the Lyapunov functions with the neural network estimates of the performance measure, making it viable for systems for which the dynamics are not known precisely or Lyapunov functions cannot be found by analysis.

3 Neural Networks

Neural networks are used in two ways in the proposed scheme. The *stability region estimator* is a classifier, identifying when the system is stable for a given state. The *performance estimator* produces an estimate of the cost-to-go function (3) from a given state. In both cases a two-layer feedforward network is used for its capacity as an universal approximator with a size that is small relative to the size of the data set [3].

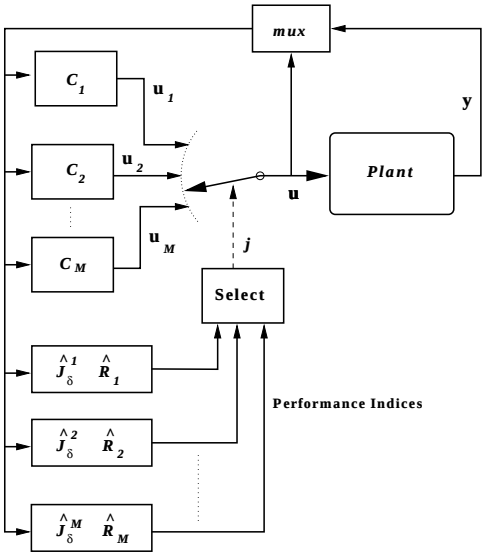

Fig. 1. Control Scheduling diagram.

The input-output behavior of the two-layer network with linear outputs units is described by

$$\mathbf{y} = \mathbf{b}_o + W_o^T \mathbf{x} + W_3^T \ \phi_2(\mathbf{b}_2 + W_2^T \ \phi_1(\mathbf{b}_1 + W_1^T \mathbf{x})), \tag{5}$$

where $\mathbf{x} \in \mathcal{R}^{n_o}$ is the input vector, $\mathbf{y} \in \mathcal{R}^{n_3}$ the output vector, $W_i \in \mathcal{R}^{n_i \times n_{i-1}}$, $i = 1, 2, 3$ and $W_o \in \mathcal{R}^{n_3 \times n_o}$, the weight matrices for each layer of n_i units, $\mathbf{b}_i \in \mathcal{R}^{n_i}$ the threshold vectors for each layer and $\phi_i : \mathcal{R}^{n_i} \rightarrow \mathcal{R}^{n_i}, i = 1, 2$; the nonlinear functions for each hidden layer. The functions $\phi_i(.)$ for the hidden units of the neural network are all chosen to be the hyperbolic tangent functions applied to each component of its input vector (i.e. $\phi_{ij}(\mathbf{x}) = \tanh(\mathbf{x}_j)$). In our application, the input vector $\mathbf{x}$ is the state of the plant for both the stability region estimators and performance estimators.

3.1 Estimating Stability Regions

For the stability region estimators, the neural network output $\mathbf{y}$ is a two-dimensional vector with components ranging roughly between -1 and 1 in the region of approximation. The ideal output values are $(y_1, y_2) = (1, -1)$ when $\mathbf{x}$ belongs to R^i and $(y_1, y_2) = (-1, 1)$ when $\mathbf{x}$ is not in R^i. To make a distinct classification in the non-ideal case (i.e., when the components of $\mathbf{y}$ are not equal to ± 1), two positive threshold parameters, θ and δ, are selected to implement the following decision rule:

Stability Region Classifier: Declare $\mathbf{x}$ belongs to R^i if and only if:

$$\begin{array}{l} 1.\ y_1(\mathbf{x}) - y_2(\mathbf{x}) > \theta, \quad \text{and} \\ 2.\ \|\nabla_{\mathbf{x}}(y_1(\mathbf{x}) - y_2(\mathbf{x}))\| < \delta, \end{array}$$

where the notation $\nabla_{\mathbf{x}}$ denotes the gradient with respect to $\mathbf{x}$.

The stability region classifier is motivated by the necessity of obtaining conservative approximations for the stability regions. The parameter θ is chosen after the network is trained so that the classification is correct for all the training and validation data. The parameter δ is chosen much smaller than the maximum of the norm of the gradient of the network output over the domain.

The training of the neural network for the stability region estimator is based on supervised learning procedures. This approach is widely used for pattern recognition and classification applications [3]. To initialize the training for each controller, the following three regions $A \subset B \subset C$ are defined, based on a priori knowledge of the closed-loop system behavior:

1. *Inner region A.* A very conservative region which includes all the states from which covergence to the origin is certain.

2. *Study region B.* The region on which the training procedure is going to be conducted.

3. *Unsafe region C.* The bounding region in which the system is either unstable or the state is outside the operating region of interest.

By making experiments with the controllers starting at states belonging to region B, and observing if the evolution leads the system to region A or to region C, data is obtained for region B to train the neural network. This procedure is carried out initially using off-line data, but training can continue on line as the system operates.

Comparisons of the neural network stability region estimator with other approaches to stability region approximation have been presented in [7]. We have found that in all cases, with a reasonable amount of training, the neural network obtains an estimate of the stability region which is much less conservative than most other methods. Moreover, since it is not model-based, the neural network classifier can be applied to systems using empirical data.

3.2 Estimating Performance Indices

Estimating performance indices such as (3) is a standard problem in Neuro-dynamic programming [2]. A Heuristic Dynamic Programming (HDP) algorithm [13] is used to train the networks. The training algorithm uses an estimation of the cost-to-go at $\mathbf{x}_k$ given by

$$J_\delta^{i*}(\mathbf{x}_k) = U^i(\mathbf{x}_k) + \delta \hat{J}_\delta^i(\mathbf{x}_{k+1}) \tag{6}$$

where $J_\delta^{i*}(\mathbf{x}_k)$ is the desired value for the network for state $\mathbf{x}_k$. Equation (6) is motivated by the definition of $J_\delta^i(\mathbf{x})$ (3) neglecting the bias term B^i

which is added directly to the output of the network. In our applications, $U^i(\mathbf{x})$ is a standard quadratic form,

$$U^i(\mathbf{x}) = \mathbf{x}^T\ P\ \mathbf{x}, \qquad P^T = P > 0,\ i = 1, \ldots, M.$$

The HDP training procedure, illustrated in figure 2, is described briefly as follows. Given the new state value $\mathbf{x}_{k+1}$ at time $k+1$, the neural network with parameters from time k, denoted $\mathrm{NN}_i(k)$, is used to predict both $\hat{J}^i_\delta(\mathbf{x}_k)$ and $\hat{J}^i_\delta(\mathbf{x}_{k+1})$. The latter value is used to compute $J^{i*}_\delta(\mathbf{x}_k)$ as defined in (6). The difference $\epsilon_k = J^{i*}_\delta(\mathbf{x}_k) - \hat{J}^i_\delta(\mathbf{x}_k)$ is used, in a back-propagation algorithm, to update the parameters in the neural network to produce $\mathrm{NN}_i(k+1)$ (indicated by the arrow through the NN_i block).

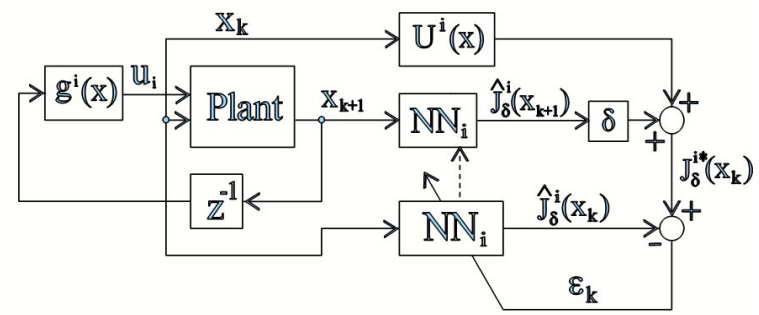

Fig. 2. HDP learning scheme.

Analytical results for related problems in the context of Q-learning [5] and temporal differences [12] indicate that convergence should be expected under rather mild conditions. A principal difference between our application and most work on learning cost-to-go performance indices is that the estimated values of the performance indices do not influence the control laws. We assume, rather, that each of the given controllers stabilizes the system and the feedback laws remain fixed.

4 Analysis of Closed-Loop Performance

We first consider the system behavior in the perfect information case, that is, when the performance measures and stability regions are known for each controller. We then consider the effect of using the neural network estimators rather than the exact values.

The approach to analyzing the closed-loop system follows the technique for the min-switching strategy suggested in [8] for the continuous-time case. To restate the basic Lyapunov results for our discrete-time context, suppose for each control law $\mathbf{g}^i$ there is a known Lyapunov function $V^i(\mathbf{x})$ for the closed-loop system under that control law within the region of

stability R^i. Moreover, suppose the control applied at each sample instant is chosen according to the min-switching strategy, that is, the control is selected which corresponds to a Lyapunov function with the minimum value among all the Lyapunov functions evaluated at the current state. The following theorem is the discrete-time version of the result in [8].

Theorem 1. *If the system given by* (1) *is controlled by the min-switching strategy applied to a set of known Lyapunov functions, the origin is asymptotically stable in the region* $R = \bigcup R^i$. *Moreover, the function*

$$W(\mathbf{x}) = \min_{i \in \{j \mid \mathbf{x} \in R^j\}} \{V^i(\mathbf{x})\} \tag{7}$$

is a Lyapunov function on R.

Proof. Follows from the continuous-time result in [8], *mutatis mutandis*.

We now apply this result to the min-switching strategy considered in this paper by observing that if the performance indices J_1^i (3) converge ($\delta = 1$), they are in fact Lyapunov functions for the respective controllers.

Theorem 2. *Given the system defined by* (1) *and a collection of control laws* $\mathbf{g}^i, i = 1, \ldots, M$. *Suppose for each control law the origin is asymptotically stable for the closed-loop system*

$$\mathbf{x}_{k+1} = F^i(\mathbf{x}_k)$$

in a connected region R^i, *and* J_δ^i *given by* (3) *converges for* $\delta = 1$. *Then there exists a* $\delta^* \in (0,1)$ *such that the origin is asymptotically stable in the region* $R = \bigcup R^i$ *for the closed-loop system controlled by the min-switching strategy [8] for any* $\delta \in (\delta^*, 1]$. *Moreover, for any* $\delta \in (\delta^*, 1]$ *the function*

$$J_\delta = \min_{i \in \{j \mid \mathbf{x} \in R^j\}} \{J_\delta^i\} \tag{8}$$

is a Lyapunov function on R.

Proof. For each i, if J_δ^i converges, it follows from the definition of J_δ^i that it is continuous in δ and therefore there exists some $\delta_i^* \in (0,1)$ such that for all $\delta \in (\delta_i^*, 1]$ J_δ^i is a Lyapunov function for the closed-loop system under control law i on R^i. The theorem follows by letting $\delta^* = max(\delta_1^*, \ldots, \delta_M^*)$.

We now turn to the min-switching strategy using the neural network estimators. In the following we assume the stability region estimators

are all conservative, that is, for all $i = 1, \ldots, M$, $\hat{R}^i \subseteq R^i$. Moreover, we assume all the stability region estimates are nonempty and connected. These assumptions are reasonable given the properties of neural network classifiers and the ability to initiate the training for the stability region estimators based on a priori knowledge of the capabilities of the given controllers.

Theorem 3. *Suppose the assumptions of Theorem 2 are satisfied and the performance estimates are computed using $\delta \in (\delta^*, 1]$ where δ^* is as specified in Theorem 2. Furthermore, suppose J^i_δ is continuous on R^i. If for some $\epsilon > 0$ the performance estimates satisfy*

$$|\hat{J}^i_\delta(\mathbf{x}) - J^i_\delta(\mathbf{x})| < \epsilon \quad \text{for all } \mathbf{x} \in \hat{R}^i, \ i = 1, \ldots, M \tag{9}$$

and the min-switching strategy is applied for some $\mathbf{x}_o \in \hat{R} = \bigcup \hat{R}^i$ resulting in a state trajectory such that there exists $K \in \mathcal{N}$ for which $\mathbf{x}_k \in \bigcap \hat{R}^i, \forall k > K$, then $\mathbf{x}_k \to X_\epsilon$ where

$$X_\epsilon = \bigcup_i X^i_\epsilon \ = \ \bigcup_i \{\mathbf{x} \mid J^i_\delta(\mathbf{x}) \leq \sup_{\tilde{\mathbf{x}} \in \chi^i_\epsilon} J^i_\delta(\tilde{\mathbf{x}})\} \tag{10}$$

and

$$\chi^i_\epsilon = \{\mathbf{x} \in R^i \mid -\Delta J^i_\delta(\mathbf{x}) \leq 2\epsilon\}.$$

Proof. For a given $\mathbf{x}_o \in \hat{R}$, let i_k be the sequence of controllers selected by the min-switching rule. If there exists some K and $l \in \{1, \ldots, M\}$ such that $i_k = l$ for all $k > K$, the theorem is true since the origin is a stable equilibrium for controller l. On the other hand, if the controller switches infinitely often, there must be one controller l which is selected infinitely often. Let the sequence of time indices $0 < k_1 < k_2 \ldots$ be an infinite sequence of sampling instants when controller l is selected with $\mathbf{x}_k \in \bigcap \hat{R}^i, \ \forall \ k > k_1$. The min-switching rule implies

$$\hat{J}^l_\delta(\mathbf{x}_{k_{j+1}}) < \hat{J}^l_\delta(\mathbf{x}_{k_j}), \quad \forall j$$

because of the limits L_{k_j}. Since the $\hat{J}^l_\delta(\mathbf{x}_{k_j})$ are bounded from below, the sequence $\hat{J}^l_\delta(\mathbf{x}_{k_j})$ converges to some constant C.

For each $i = 1, \ldots, M$ let $\eta^i(\mathbf{x})$ denote the error in the i^{th} performance estimate at state $\mathbf{x}$ where it is assumed $|\eta^i(\mathbf{x})| \leq \epsilon$ for some $\epsilon > 0$. This implies that when the i^{th} controller is applied at a state $\mathbf{x} \in \hat{R}^i$,

$$\begin{aligned} \Delta \hat{J}^i_\delta(\mathbf{x}) &\triangleq \hat{J}^i_\delta(F^i(\mathbf{x})) - \hat{J}^i_\delta(\mathbf{x}) \\ &= \Delta J^i_\delta(\mathbf{x}) + \eta^i(F^i(\mathbf{x})) - \eta^i(\mathbf{x}) \\ &\leq \Delta J^i_\delta(\mathbf{x}) + 2\epsilon. \end{aligned}$$

Since J^i_δ is a Lyapunov function for the system under controller i, $\Delta J^i_\delta(\mathbf{x}) \leq 0$. Returning to the specific controller l, suppose that there are an infinite number of the x_{k_j} that remain a finite distance from the set

$$\chi^l_\epsilon = \{\mathbf{x} \in R^l \mid -\Delta J^l_\delta(\mathbf{x}) \leq 2\epsilon\}.$$

This would imply the sequence $\Delta \hat{J}^l_\delta(\mathbf{x}_{k_j})$ is negative and bounded away from zero infinitely often, contradicting $\hat{J}^l_\delta(\mathbf{x}_{k_j}) \to C$. Therefore, $\mathbf{x}_{k_j} \to X^l_\epsilon$. More precisely, given any $\tilde{\epsilon} > 0$, there exists some $K^l_{\tilde{\epsilon}}$ such that the distance $d(\mathbf{x}_{k_j}, \chi^l_\epsilon) < \tilde{\epsilon}$ for all $k_j > K^l_{\tilde{\epsilon}}$. This is illustrated in figure 3.

While controller l is applied, $J^l_\delta(\mathbf{x}_k)$ is monotone non-increasing since J^l_δ is a Lyapunov function for the system. Therefore, $J^l_\delta(\mathbf{x}_k) \leq J^l_\delta(\mathbf{x}_{k_j})$, for $k \geq k_j$ until another controller becomes active (see figure 3). Define $I_l = \{k \in \mathcal{N} | i_k = l\}$ and $\bar{J}^l_\delta = \sup_{\tilde{\mathbf{x}} \in \chi^l_\epsilon} J^l_\delta(\tilde{\mathbf{x}})$. Then, for any given $\beta > 0$ we have

$$J^l_\delta(\mathbf{x}_k) \leq \sup_{k_j} J^l_\delta(\mathbf{x}_{k_j}) \leq \bar{J}^l_\delta + \beta \qquad \forall k \in I_l, k > K^l_{\tilde{\epsilon}}$$

because of the continuity assumption on $J^l_\delta(\mathbf{x})$. Since this has to be true for any l, after a finite K in which all the controllers that are not used infinite number of times do not become active anymore, the sequence $\mathbf{x}_k$ is arbitrarily close to X_ϵ.

This theorem indicates that when the performance estimates are used rather than the exact performance indices, the min-cost switching strategy will drive the state to a neighborhood of the origin determined by the magnitudes of the errors in the performance estimates for each controller. Theorem 3 does not guaranteed the neighborhood is arbitrarily small, however. Moreover, the exact performance measures are not known in general, so the neighborhood in Theorem 3 could not be computed even if the bound on the estimation error was known. These difficulties are eliminated when $\delta = 1$, however, since in this case we have $\Delta J^i_\delta(\mathbf{x}) = -U(\mathbf{x})$.

Corollary 4. *Under the assumptions of Theorem 3 with $\delta = 1$, if the min-switching strategy is applied for some $x_o \in \hat{R} = \bigcup \hat{R}^i$ resulting in a state trajectory such that $x_k \to \bigcap \hat{R}^i$, then*

$$x_k \to \{x \in R^n | U(\mathbf{x}) \leq 2\epsilon\}.$$

5 An Application

One of the principal motivations for developing the controller switching strategy presented in this paper is to provide a method for implementing the switching rules in the SIMPLEX architecture, a real-time environment developed at the Software Engineering Institute at Carnegie Mellon University that provides protection against errors in control system upgrades. Figure 4 shows a typical configuration for SIMPLEX in which there are three controllers: a *safety* controller, a reliable *baseline* controller, and

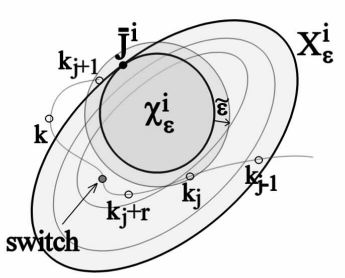

Fig. 3. Trajectory of the system illustrating convergence to X_ϵ.

an *experimental* controller representing a new, untested control module. The basic idea of the SIMPLEX system is to guarantee that the baseline controller performance is maintained if there are problems in the experimental controller. This is accomplished by monitoring the control outputs and system performance when the experimental controller is installed, and switching control back to the baseline controller if problems are detected. The safety controller is invoked when it is necessary to take more extreme action to return the system to the operating region for the baseline controller.

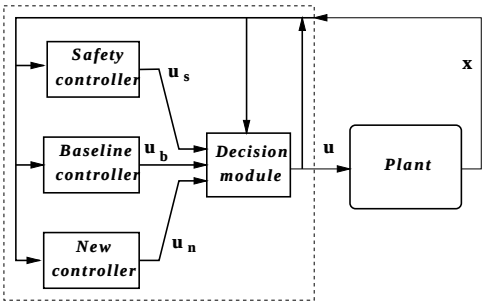

Fig. 4. SIMPLEX architecture.

Clearly the ability for the SIMPLEX system to provide the desired protection against errors in the experimental controller depends entirely on the rules used to switch between controllers. These rules are very difficult to create and maintain, even for small systems. The neural network approach proposed in this paper provides a means of obtaining less conservative estimates of the stability regions for the controllers, and also a method for determining when to switch from the safety controller back to the baseline controller based on estimates of their performance.

We present results here on the implementation of the *min-switching* con-

trol strategy for a level-control system for an underwater vessel. This system has been designed in the Software Engineering Institute at Carnegie Mellon University as a testbed for the development of dependable and evolvable systems using the SIMPLEX architecture. The experimental system consists of a water tank in which a vessel can move vertically by changing the size of the air bubble inside it. Air is moved in and out of the vessel through a flexible tube connected to a cylinder-piston mechanism. Figure 5 shows a schematic diagram of the system components. The control goal is to stabilize the vessel at an arbitrary position inside the water tank. The position of the vessel and the size of the air bubble are measured directly using ultrasound sensors. A stepper motor controls the piston movement. Constraints are imposed by the bottom of the tank and the water level. The control input is limited by the maximum speed of the stepper motor.

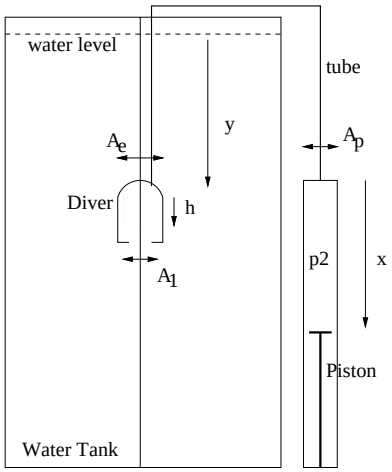

Fig. 5. Schematic diagram for the submerged vessel system.

A set point of $y_{st} = 25\ in$ was selected. Two controllers were used to test the switching strategy. Both controllers are state feedback controllers used in the original SIMPLEX architecture implementation. One controller, $\mathbf{u}_1$, has an acceptable performance close to the set point while the second controller, $\mathbf{u}_2$, performs better in a larger operating region using a bang-bang action, but with unacceptable oscillations near the set point. An analytical model was used for initial training of the neural network, then experimental data from twenty runs were used to adapt the parameters of the neural networks to estimate the performance indices of both controllers. Figure 6 shows a typical data profile to estimate the performance index of one of the controllers and figure 7 shows a slice of the resulting performance estimate.

Figures 8 shows a switching experiment for a step change in the setpoint

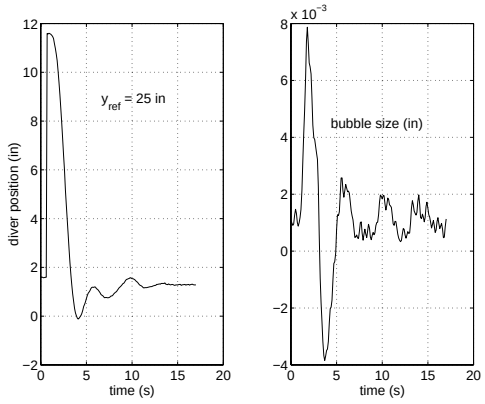

Fig. 6. Data obtained from an experimental run for a controller.

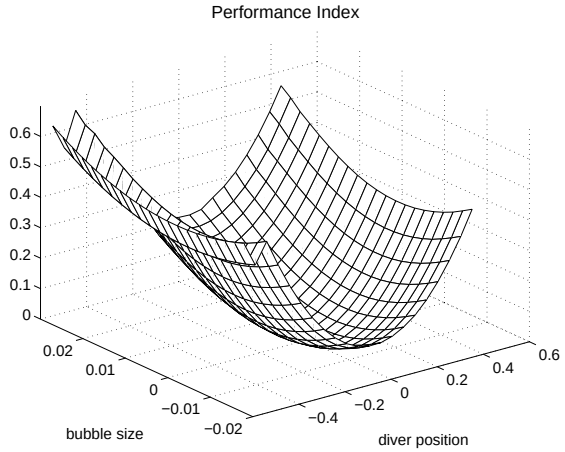

Fig. 7. Performance index estimate ($y = 0$).

value for y_{st} from 13 to 25 inches. Figure 9 shows the estimated performance indices during the run. From the figure we observe that controller 2 is preferred for larger values of y. After approximately 5.5 seconds $\hat{J}_\delta^1$ becomes smaller than $\hat{J}_\delta^2$ and the scheduler switches to controller 1.

6 Discussion

This paper presents a method for switching among a set of given controllers using multilayer feedforward neural networks and neuro-dynamic programming techniques. In contrast to switching control strategies aimed

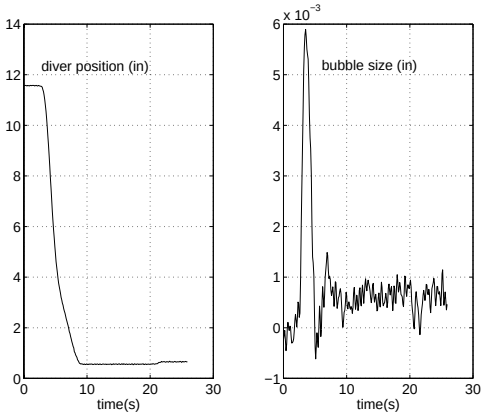

Fig. 8. Level relative position of the vessel and bubble size during a switching experiment.

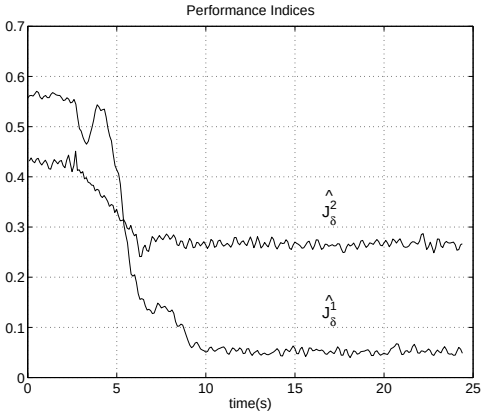

Fig. 9. Performance indices estimates for controllers for the submerged vessel during a switching experiment.

at adapting to unknown plant dynamics, the objective in this work is to select the best controller from among a set of controllers that have been designed for a known plant. This objective is most closely aligned with the switching control strategies proposed in [8] and [6]. By using neural networks to estimate the stability regions and performance indices for the controllers, the switching strategy depends on experimental data from the actual system, rather than analytical models that may lead to misleading or incorrect switching rules. We present a new result on the stability of the min-switching strategy using estimates of Lyapunov functions.

The convergence and stability results in this paper are sufficient con-

ditions. There are several open problems concerning the verification of these conditions in applications and the possibility of obtaining less conservative results. For the closed-loop behavior, the ramifications of continued learning and persistent excitation need to be studied further. It would be desirable to introduce techniques by which performance estimate learning could be achieved for the controllers that are not currently controlling the system, by introducing, perhaps, a model of the system being controlled. The introduction of adaptive control to deal with changes in the plant dynamics may also be important for some applications.

Acknowledgments

This research has been supported by the Uruguayan government through a CONICYT-IBD grant, the Organization of American States and DARPA, contract number F33615-97-C-1012.

References

1. J. Balakrishnan and K.S. Narendra. Adaptive control using multiple models. *IEEE Trans. on Automatic Control*, 42(2):171–187, Feb 1997.

2. Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

3. Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, Great Britain, 1995.

4. M. Bodson, J. Lehoczky, R. Rajkumar, L. Sha, D. Soh, M. Smith, and J. Stephan. Control configuration in the presence of software failures. In *Proc. 32nd IEEE Conf. Decision Control*, volume 3, page 2284, San Antonio, TX, Dec 1993.

5. S.J. Bradtke, B.E. Ydstie, and A.G. Barto. Adaptive linear quadratic control using policy iteration. In *Proc. American Control Conference*, volume 3, pages 3475–9, Baltimore, MD, Jun 1994.

6. Michael S. Branicky. Stability of switched and hybrid systems. In *Proc. 33rd IEEE Conf. Decision Control*, volume 4, pages 3498–3503, Lake Buena Vista, FL, Dec 1994.

7. E.D. Ferreira and B.H. Krogh. Using neural networks to estimate regions of stability. In *Proc. of 1997 American Control Conference*, volume 3, pages 1989–93, Albuquerque, NM, Jun 1997.

8. J. Malmborg, B. Berhardsson, and K.J. Aström. A stabilizing switching scheme for multi-controller systems. In *Proc. of the IFAC*

World Congress, volume F, pages 229–234, San Francisco, California, USA, 1996. IFAC'96, Elsevier Science.

9. A.S. Morse. Supervisory control of families of linear set-point controllers - part i: Exact matching. *IEEE Trans. on Automatic Control*, 41(10):1413–31, Oct 1996.

10. D. Seto, L. Sha, A. Chutinan, and B.H. Krogh. The simplex architecture for safe on-line control system upgrades. Submitted to 1998 American Control Conference.

11. L. Sha. A software architecture for dependable and evolvable industrial computing systems. In *Proc. IPC'95*, pages 145–156, Detroit, MI, May 1995.

12. J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Trans. on Automatic Control*, 42(5):674–690, May 1997.

13. P. Werbos. A menu of designs in reinforcement learning over time. In W.T. Miller III, R.S. Sutton, and P. Werbos, editors, *Neural Networks for control*, chapter 3. MIT Press, 2nd edition, 1991.

This article was processed using the LaTeX macro package with LLNCS style