Controller Scheduling Using Neural Networks: Implementation and Experimental Results

Enrique D. Ferreira and Bruce H. Krogh
Department of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Av.
Pittsburgh, PA, 15213-3890 USA
edf/krogh@ece.cmu.edu

Submitted to *Springer Lectures Notes on Computer Science*
Special Issue on Hybrid Systems: Hybrid Systems V

Abstract

This paper presents the results of simulation and control experiments using a recently proposed method for real-time switching among a pool of controllers. The switching strategy selects the current controller based on neural network estimates of the future system performance for each controller. This neural-network-based switching controller has been implemented for a simulated inverted pendulum and a level control system for an underwater vehicle in our laboratory. The objectives of the experiments presented here are to demonstrate the feasibility of this approach to switching control for real systems and to identify techniques to deal with practical issues that arise in the training of the neural networks and the real-time switching behavior of the system. This experimental work complements on-going theoretical investigations of the method which will be reported elsewhere.

1 Introduction

Recently there has been an interest in switching control schemes for adaptive control [9], [10], [2], [1], and fault-tolerant control [11]. A switching control scheme based on neural network estimates of performance indices for the candidate controllers was proposed in [6]. The objectives of theoretical investigations of switching controllers are to establish sufficient conditions for closed-loop stability and bounds on closed-loop performance. This paper reports on empirical investigations of the switching control scheme proposed in [6] and considers some of the practical issues that arise in the implementation of switching control schemes. Some theoretical results for this particular switching strategy have been reported elsewhere [5].

The basic rule of the switching strategy is to select the controller at each sampling time with the lowest performance index, as estimated by the ensemble of neural networks. This strategy is motivated by the so-called min-switch rule for multiple Lyapunov functions [4, 8].

Each of the performance indices being estimated is a cost-to-go function which, under mild assumptions, would be a Lyapunov function for the system under control of the corresponding state-feedback law. If these Lyapunov functions were known precisely, the min-switching strategy would lead to an asymptotically stable system. The strategy investigated in this paper addresses the practical issue of how to implement the min-switching strategy for real systems. For real systems, the true dynamics and Lyapunov functions are never known exactly. We demonstrate empirically that neural networks can be used to estimate cost-to-go functions effectively, and that acceptable closed-loop asymptotically stable behavior can be achieved using these estimates.

This paper focuses on performance-based switching. The complete switching strategy includes an estimate of the region of stability for each controller; a controller is not available if the current state is outside the region of stability for the closed-loop system under that controller. Results related to the estimation of regions of stability using neural networks are presented in [7]. Typically this min-switching rule avoids selecting controllers for which the state of the system is outside the stability region, however. Since the performance index is very large (possibly infinite) for the controllers for which the state is outside the stability region, we found the stability region estimates to be unnecessary in the experiments reported in this paper.

The paper is organized as follows. The next section presents a real-time infrastructure for fault-tolerant control as a primary motivation for the switching control problem considered in this paper. Two applications are also introduced for which multiple controllers have been designed and the problem is to design switching rules to select the appropriate controllers in real time. Section 3 presents the neural-network-based switching strategy. Section 4 describes the method for training the neural networks to estimate closed-loop performance based on the concept of neural dynamic programming. Data is presented for training the neural networks for the two example applications. We then consider the effectiveness of the switching control strategy in section 5 where experimental results are presented for the two experimental examples. It is observed that chattering can occur due to the errors in the neural network estimates and methods for eliminating the chattering are discussed. The concluding section summarizes our experience with the neural-network-based switching control and identifies several directions for further investigations, both experimental and theoretical.

2 Applications of Switching Control Strategies

Switching control strategies are generally considered for situations where there are multiple operating regimes for a system and different controllers are designed for each operating regime. Switching control has also been proposed as an approach to adaptive control to deal with unknown plants. In this paper we consider a version of the former scenario where different controllers are available to stabilize a given system, but with different performance objectives. One controller might be designed to provide good transient performance for small deviations from the equilibrium, whereas another controller may be designed to drive the system quickly back to the equilibrium when large deviations are detected. The latter controller may perform poorly when the system is near the equilibrium, so it is desirable to switch back to the first controller at some point.

In this section we first describe a real-time infrastructure that relies on switching control strategies to provide fault tolerance against errors in untested control code. This architecture provides a practical motivation for the switching control strategy proposed in [6] and is the platform used for the control experiments presented in this paper. We then present two applications for which multiple controllers are available as test cases for the empirical

investigations.

2.1 SIMPLEX Architecture

One of the main 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 technology developed at the Software Engineering Institute at Carnegie Mellon University to support safe, reliable on-line upgrades and modifications to real-time control systems [11].

Figure 1 shows a basic configuration for SIMPLEX. It consists of three controllers: a *safety* controller, a reliable *baseline* controller, and the *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 with the experimental controller. This is accomplished by monitoring the control outputs and system performance when the experimental controller is operating on the system, 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. The controllers run concurrently as separate tasks in a real-time multitasking operating system, or even on different machines. SIMPLEX is in charge of all the communication between the parts of the system and monitors the performance of the controllers acting on the plant to determine which controller should actually be controlling the plant at each instant.

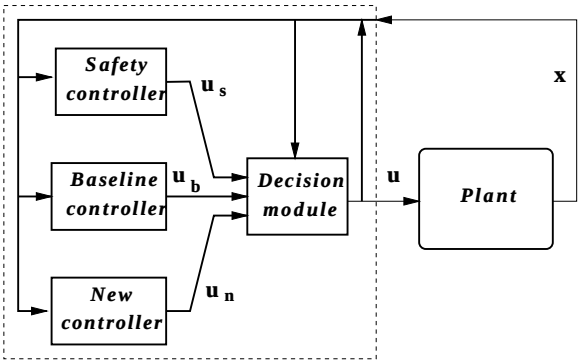

Figure 1: SIMPLEX architecture.

The SIMPLEX Architecture has been applied successfully to the control of several physical systems including single and multiple inverted pendulum systems, a plasma deposition system, and a level-control system for a submerged vessel. We present two of these systems in the following subsections. From a control perspective, SIMPLEX corresponds precisely to the multi-controller scenario considered in this paper. The controllers are designed for a given plant with similar control objectives, in most cases to stabilize the system, but with different regions of stability and different performance characteristics. 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 to the safety and baseline controllers. These rules are very difficult to determine analytically, even for low-dimensional systems. In the current applications of SIMPLEX, physical insight and extensive experimentation have been used to create ad hoc rules for switching to the safety and baseline controllers. The neural network approach proposed in this paper provides a means for determining automatically, and with some theoretical motivation, when to switch to the safety controller and the baseline controller. The on-line learning capabilities of the neural networks also suggest the possibility of acquiring knowledge of the performance of an experimental controller, leading to the eventual transition of the

experimental controller to the new baseline controller once it has been completely tested and verified.

2.2 An Inverted Pendulum

As one example of an application of controller scheduling we use the well known inverted pendulum (IP) with a set of three controllers. One reason for using this application is that a physical IP has served as a standard laboratory example for the SIMPLEX architecture.

The complete IP model and control design is described in [6]. We include here a brief summary. The IP model equations used are:

$$\begin{aligned} J_t\ddot{x} + \frac{1}{2}\,ml\,cos\theta\,\ddot{\theta} + B_r\dot{x} - \frac{1}{2}\,ml\,sin\theta\,\dot{\theta}^2 &= F \\ \frac{1}{2}\,m\,cos\theta\,\ddot{x} + \frac{1}{3}ml\ddot{\theta} - \frac{1}{2}\,mg\,sin\theta &= 0 \end{aligned}$$

with the following constraints:

$$|F| \leq F_{max}, \quad |x| \leq x_{max}, \quad |\dot{x}| \leq v_{max}$$

where θ is the angle of the pendulum, x is the position of the "cart" (the pendulum base), m is the effective mass at the end of the pendulum, l is the pendulum length, J_t is the inertia of the cart, B_r is the coefficient of friction for the cart, and g is gravitational acceleration.

The parameters for the physical system in the laboratory are:

J_t	0.6650	Kg	g	9.8	m/s^2
m	0.21	Kg	F_{max}	2	N
l	0.61	m	x_{max}	2.0	m
B_r	0.1	Kg/s	v_{max}	3.0	m/s

The controllers designed for the system are: a standard linear quadratic regulator (LQR) controller for the linearized model of the IP around the origin in the full state space, a velocity feedback (VF) controller that takes the pendulum angle θ and the velocity of the cart to zero (but ignores the cart position), and a sliding mode (SM) controller designed to bring θ to zero as fast as possible using a nonlinear switching rule and saturating control (but without regard to the cart position or the cart speed).

The need for a switching strategy is clear. There are different controllers with different characteristics available for a physical system. Given the designs of these three controllers, one would expect the performance of the SM controller to be best for larger pendulum angles, the VF controller to work well when the position of the cart is large, and the LQR controller to provide the best performance for small deviations from the origin of the four-dimensional state space. Therefore, there is a need for a supervisor to schedule those controllers in the best way possible, where "best" means asymptotic stability with good transient performance.

2.3 Submerged Vessel

This application is an experimental system in our laboratory consisting of a water tank in which a vessel ("diver") 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 2 illustrates the system components and figure 3 lists the physical parameters for the system. 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. The control objective is to stabilize the vessel at an arbitrary level without allowing the vessel to touch the bottom of the tank or emerge above the surface of the water.

Any equilibrium position for the vessel is open-loop unstable since the air bubble compresses(expands) as the vessel goes down(up). There is also a significant time delay in the response of the vessel level to changes in the cylinder position due to the volume of the tube connecting the piston with the vessel. The following model has been developed for the system using first principles,

$$\begin{aligned} \ddot{y} &= -a|\dot{y}|\dot{y} - bh - f_r(\dot{y}) \\ \dot{h} &= -\alpha(y,h) - \beta(y,h)u(t-\tau) \end{aligned}$$

with

$$\begin{aligned} f_r(\dot{y}) &= f_1\dot{y} + f_2\ sgn(\dot{y})\ e^{-f_3|\dot{y}|} \\ \alpha(y,h) &= \frac{\alpha_1}{\alpha_2 + (\alpha_3 + \alpha_y y + \alpha_h h)^2} \\ \beta(y,h) &= \frac{(\beta_1 + \beta_y y + \beta_h h)^2}{\alpha_2 + (\alpha_3 + \alpha_y y + \alpha_h h)^2} \end{aligned}$$

where y is the position of the vessel with respect to the water level and h is the height of the bubble. For the first equation governing y, the position of the vessel, the first term of the right hand side is the water resistance, the second term reflects Archimedes' principle, and the last term is the Newtonian static and dynamic friction. The controlled input is the velocity of the piston that pushes the air between the vessel and the cylinder and τ is the time delay. State 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.

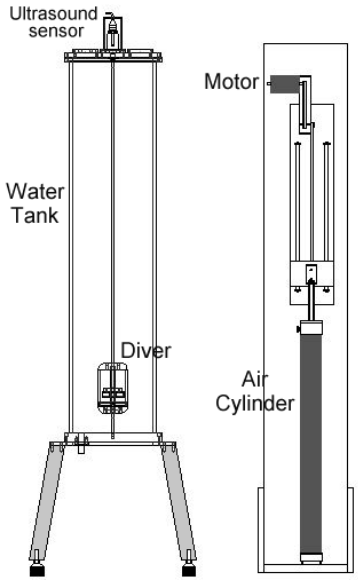

Figure 2: Sketch of the components of the submerged vessel system.

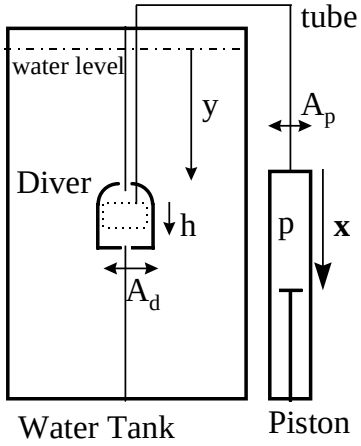

Figure 3: Physical parameters for the submerged vessel system.

For this example two linear state feedback controllers are available to regulate the position of the vessel. One of them is a bang-bang controller (BB) designed to operate when the vessel position is far from the desired position. The other controller is a linear state-feedback controller (LSF) that operates best when the vessel is close to the setpoint. The first controller drives the vessel to the setpoint much more quickly than the second controller, but the vessel

oscillates around the setpoint in a limit cycle if the first controller is applied indefinitely. The scheduler must select which controller to apply at each instant to drive the vessel to the setpoint with the best performance, which includes some measure of the speed at which the vessel reaches the equilibrium and the quality of the regulation once the vessel is near the setpoint.

3 Neural Network Based Switching Strategy

As a general formulation of the problem being considered, let a state-constrained and control-constrained nonlinear system in its state space formulation be given by

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

where $\mathbf{x}_k \in R^n$ is the state vector and $\mathbf{u}_k \in R^m$ the control input vector. The nonlinear function f is assumed to be smooth. The discrete-time state equations reflect the sampled-data nature of the computer control system. We assume that the state $\mathbf{x}_k$ is observable. The control signal $\mathbf{u}_k$ to be applied on the system can be generated by M different state feedback controllers of the form:

$$\mathbf{u}_k^i = g_i(\mathbf{x}_k), \qquad i = 1, 2, \ldots, M. \tag{2}$$

The goal of the controllers is to take the system to the origin. We are interested in developing a strategy to select which controller should be applied to the system at each sampling instant in order to achieve stable response with good transient performance. In a similar setting there have been some results reported concerning the stability of a switching controller. Branicky [4] has established conditions for a switched autonomous system to be stable in the sense of Lyapunov using the concept of Lyapunov-like functions. Malmborg [8] examines asymptotic stability and explained chattering as a possible behavior generated intrinsically by the switching and not just as an implementation problem.

To define our scheduling rule we need to define a performance index for each autonomous system generated by the application of a specific controller. Let us denote the trajectory of the system (1) with the initial state $\mathbf{x}_o$ at time step k by the application of the state feedback control $\mathbf{u}_k^i$ as $\mathbf{x}_k^i(\mathbf{x}_o)$. We consider performance indices for the controllers of the form

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

where $0 < \delta < 1$ is a discount factor and $U(.)$ represents the cost function of being in a particular state.

The proposed approach for selecting the controller at each sampling instant is illustrated in figure 4. Neural networks are used to estimate the performance indices J_δ^i at the current state, denoted by $\hat{J}_\delta^i$. The index of the control input to be applied for the next period, denoted i_k, is then selected as

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

where

$$I(\mathbf{x}_k, L_k) = \{i | i = i_{k-1} \text{ or } i \neq i_{k-1}, \hat{J}_\delta^i(\mathbf{x}_k) < L_k^i\} \tag{5}$$

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

$$L_k^i = \begin{cases} L_{k-1}^i & \text{if } i \neq i_k \\ \min\{\hat{J}_\delta^i(x_k), L_{k-1}^i\} & \text{if } i = i_k \end{cases} \tag{6}$$

Expressed in words the switching rule will select the controller with the best performance but from a restricted set of controllers called $I(\mathbf{x}_k, L_k)$ consisting of the current controller and all the other controllers which have a better performance estimate than the last time they were used. This selection rule is related to the conditions stated by Branicky [4] to have stability in the sense of Lyapunov. Figure 4 illustrates how the scheduling rule works. When the estimate of the performance index of another controller becomes the lowest one and it is also lower than the last time that controller was applied – represented by the variable L_k^i – the switching rule selects that controller. An augmented version of this rule may include the stability region for each controller as an additional requirement for it to be in the set $I(\mathbf{x}_k, L_k)$. It has been pointed out above that this was not necessary for the applications in this paper.

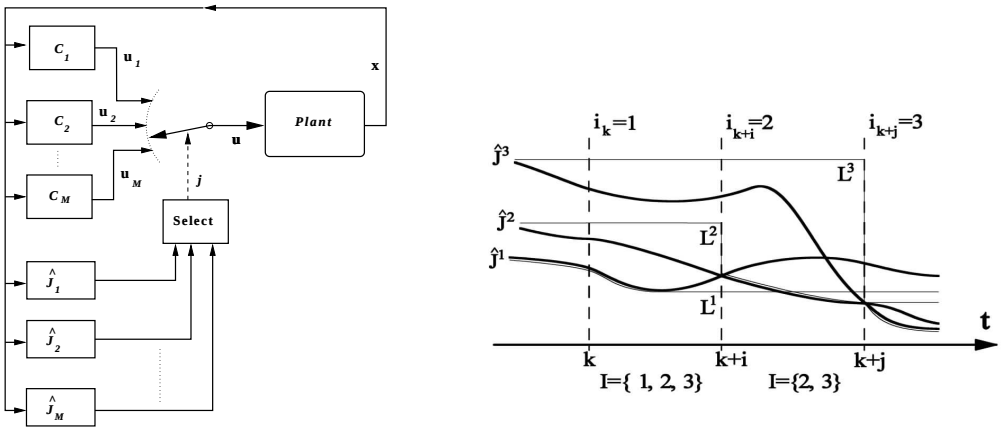

Figure 4: Controller switching scheme used and example to show how it works.

4 Neural Network Training

This section describes a neural network architecture and training techniques to estimate the index of performance for a given controller. The proposed approach is based on a modification of a dynamic programming procedure called neuro-dynamic programming (NDP) [3]. In NDP, a neural network is used to estimate the cost-to-go function. One motivation for using a neural network is for data compression: a neural network is much smaller than, say, a table representation of the cost-to-go function. The other motivation for using neural networks is the availability of techniques to train them to approximate the cost-to-go functions using data from the system.

To estimate the performance indices $\hat{J}_\delta^i(\mathbf{x})$ defined in section 3 we chose neural networks of the form:

$$\hat{J}_\delta^i(\mathbf{x}) = \mathbf{b}_o + W_o\mathbf{x} + W_3\ \phi(W_2\ \phi(W_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2), \tag{7}$$

where $W_i, \mathbf{b}_i$ are the weight matrices and threshold vectors, respectively. The nonlinear function $\phi(.)$ for the hidden units of the neural network is chosen to be the hyperbolic tangent function. The output units are selected as linear. Figure 5 shows a diagram of the network architecture.

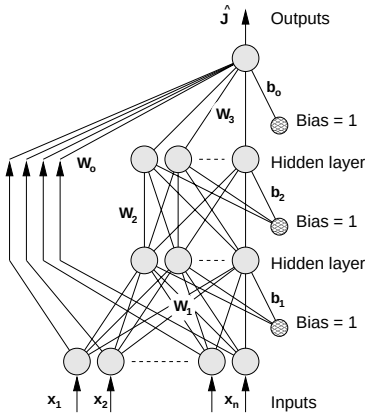

Figure 5: Multilayer neural network.

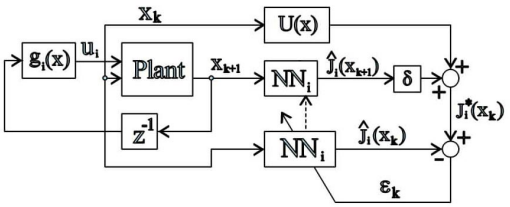

Figure 6: HDP learning procedure diagram

Neuro-dynamic programming requires a type of learning procedure that is different from the well known supervised learning algorithms. We use Heuristic Dynamic Programming (HDP) algorithm [14]. If we denote by $\hat{J}_i(\mathbf{x})$ the network estimate of the future performance for the ith controller, the learning equation takes the form

$$J_\delta^{*i}(\mathbf{x}_k) = U(\mathbf{x}_k) + \delta \hat{J}_\delta^i(\mathbf{x}_{k+1}) \tag{8}$$

where $J_\delta^{*i}(\mathbf{x}_k)$ is the desired value for the network for state $\mathbf{x}_k$. The pointwise cost function selected is a standard quadratic form:

$$U(\mathbf{x}) = \mathbf{x}^T \; P \; \mathbf{x}, \qquad P > 0$$

Using this cost function for each trajectory, the input-output patterns for the network can be computed. An error function for the network can be evaluated and a minimization procedure applied afterwards to adapt the parameters of the neural network. Figure 6 represents the learning procedure schematically.

With this procedure we characterize each autonomous nonlinear system resulting from the application of a single controller. In this sense it is different from the Q-learning approach [13] that points to the design of a new controller by generating a performance index surface resulting from the collective action of all the controllers available at each state.

In the case of the inverted pendulum example, three neural networks were trained to approximate the cost-to-go function of each closed-loop system generated by closing the loop with the three controllers. Using the simulation model, training was performed off-line in each case using 5000 random trajectories starting at states uniformly distributed over the allowable set of states. A set of 50 trajectories was used as a validation set. On-line learning is continued during the simulated control experiments using HDP. Equation (8) with $\delta = 0.5$, is used to generate the input-output patterns for the network. A square error function is computed and conjugate gradient optimization methods were used to minimize it. Off-line training was stopped when the error was 0.02 to prevent overtraining. Figures 7 and 8 show the estimation achieved for two controllers in a slice of the state space. The shapes are expected, close to a quadratic function near the origin and reaching large values outside its neighborhood. There is also some symmetry in the surface that can be explained from the model and cost functions symmetry.

For the submerged vessel system two multilayer networks were set up to learn the performance index for the two controllers. In this case, 20 experiments were run on the physical system for each controller to get the data for training the networks. It should be pointed out

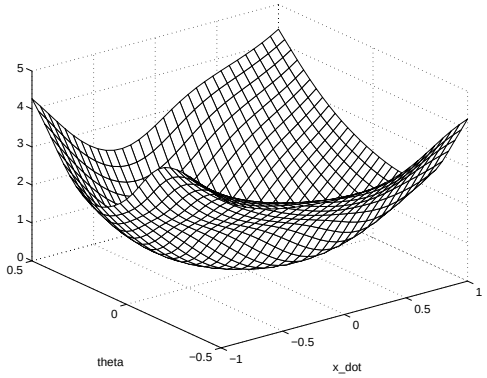

Figure 7: Performance estimate for LQR controller. ($x = \dot{\theta} = 0$)

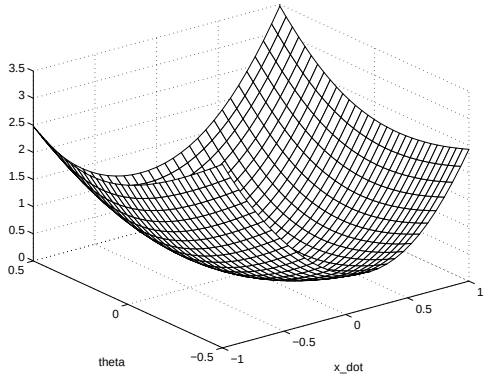

Figure 8: Performance estimate for SM controller. ($x = \dot{\theta} = 0$)

that the experiments run on the vessel were selected with starting points along the region in which we were expecting the system to operate. Five additional experimental runs were saved for network validation purposes. After that, the procedure is similar to the previous example. Overtraining criteria stopped training when the overall error was about 0.035. Figures 9 and 10 show the estimation achieve for the two controllers designed for this example in a slice of the state space. Even though the error is bigger than in the previous example the shapes of the cost-to-go estimates have similar characteristics to the ones found for the pendulum example.

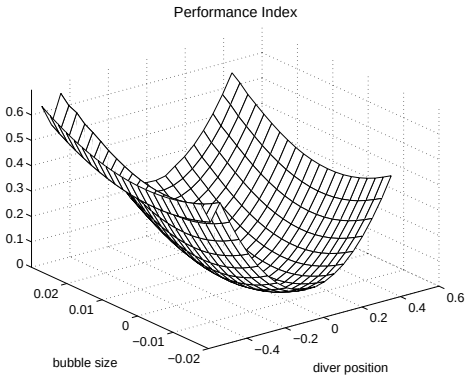

Figure 9: Performance estimate for controller LSF. ($\dot{y} = 0$)

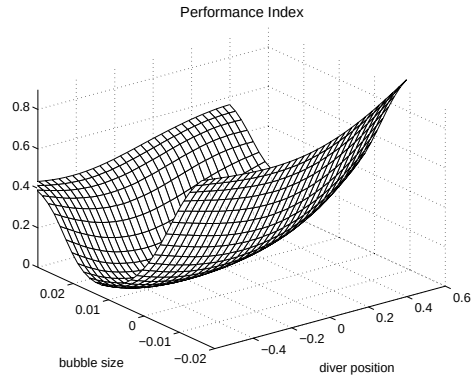

Figure 10: Performance estimate for controller BB. ($\dot{y} = 0$)

5 Real-Time Control Results

Using the IP model described in section 2.2 we tested the scheduling policy defined by equations (4), (5) and (6) with simulation. Figures 11 and 12 show the position of the pendulum and the evolution of the performance index estimates during a typical run. We see in figure 12 that the evolutions of the cost-to-go estimates are not monotonically decreasing. This is due to the fact that these are estimates of the true cost-to-go functions. Moreover, the discounted cost is not guaranteed to be a Lyapunov function for the system.

In figure 11, a slight chatter in the response of the switching mechanism can be observed. This is commonly observed in switching systems. It is caused by the discontinuity in the

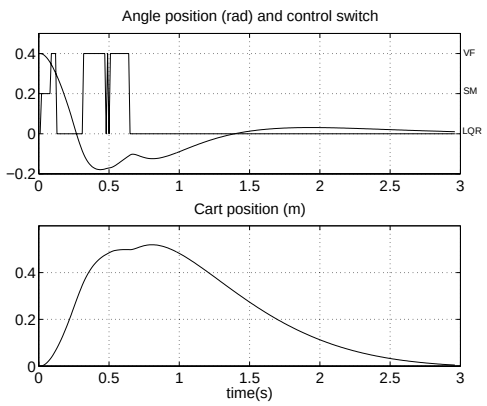

Figure 11: Switching experiment

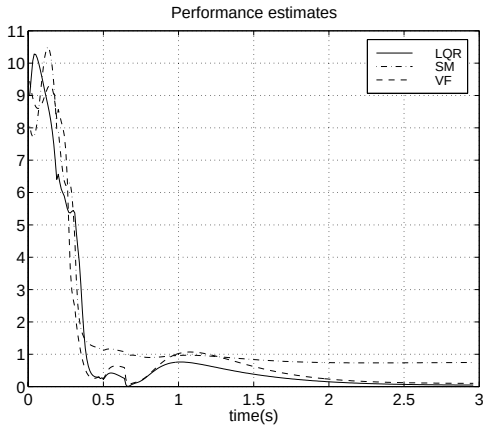

Figure 12: Performance index for each controller during run.

control signal when the system goes through the switching boundary and the new control value returns the system to the switching surface. For a strategy like the one we are using in this work, but in which the performance index is a Lyapunov function for each controller in closed loop with the system, it has been shown [8] that in certain cases a chattering control drives the system along a surface in the state space where two or more Lyapunov functions are equal. This is commonly referred to as a sliding mode. When estimates are used, chattering can also be caused by the nonmonotonicity of the performance indices.

There are several ways in which chattering has been eliminated or at least moderated before. The introduction of a boundary layer with many possible variations is a common approach [12]. Another way is the introduction of a non-zero minimum time between switches to reduce high frequency switching dynamics [2]. Another interesting idea is to compute the control that gives you the direction of the sliding surface as a way to eliminate the non-smooth behavior. The first two approaches can be applied in our method while the latter would not be possible everytime due to the usual lack of a system model. However, in this experiment the switching rule takes care of a big part of the possible chatter not allowing a controller to be used until its performance index has returned below its lowest previous value.

For the underwater vessel control system, experimental runs with the physical system where performed. Figures 13 and 14 show a switching experiment for a step change in the setpoint value for y_{st} of 12 inches. Figure 5 shows the estimated performance indices during the run. Chatter is not observed in this example. From the figure 5 we observe that the bang-bang controller 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 the linear state-feedback controller. In figure 13 we see also an offset in the final position of the vessel due to the high static friction in the vertical bar along which the vessel slides. Other controllers are being designed to be take care of that problem and can be introducing in the scheduling policy directly.

6 Summary and Discussions

A technique is presented to perform controller scheduling using a multilayer feedforward neural network and neuro-dynamic programming to estimate a performance index for each controller. Results from two applications are presented demonstrating the feasibility of the approach.

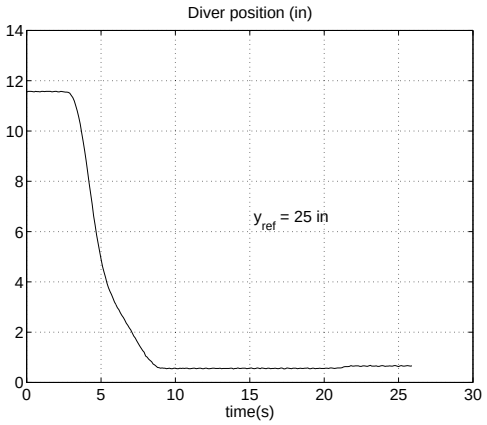

Figure 13: Level position of the vessel during a switching experiment.

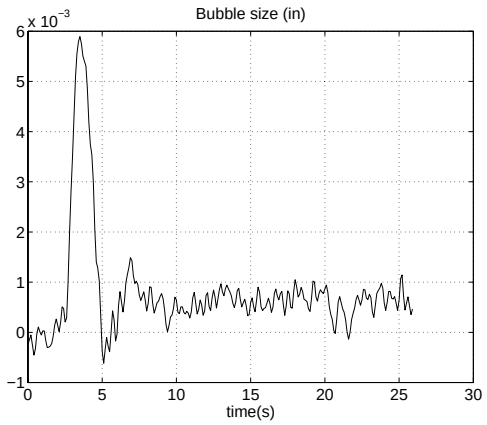

Figure 14: Bubble size during a switching experiment.

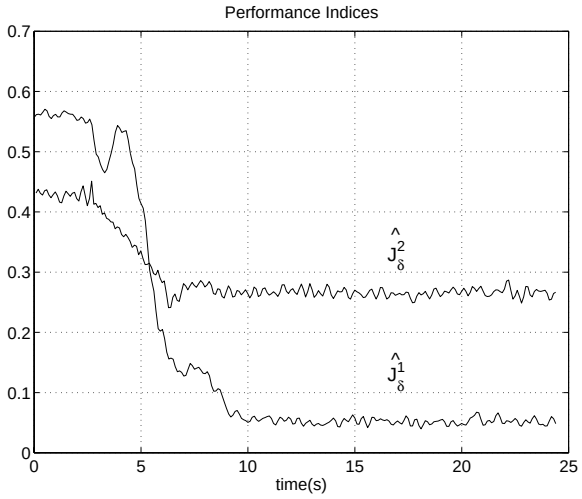

Figure 15: Performance indices estimates for controllers for the submerged vessel during a switching experiment.
$\hat{J}_\delta^1$ - LSF controller. $\hat{J}_\delta^2$ - BB controller.

Current research is focusing on learning algorithms to reduce the effect of the estimation error on the closed loop behavior. We are also investigating the use of better features as inputs to the network and different network architectures to speed up learning. Furthermore, looking at the overall picture, comparisons are being made with other switching schemes like model-based Lyapunov function scheduling techniques in terms of stability, smoothness and robustness range.

Finally, there are several theoretical issues currently under investigation. A recent result by Malmborg et al. [8] on the stability of the so-called *min-switching* rule based on Lyapunov stability theory suggests a way to analyze this architecture. Preliminary work in this area has been done but it remains under investigation to guarantee the convergence for the neuro-estimation learning algorithms and stability of the closed-loop system with adaptive controller scheduling.

7 Acknowledgments

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

References

[1] J. Balakrishnan and K.S. Narendra. Adaptation and learning using multiple models, switching, and tuning. *IEEE Control Systems Magazine*, 15(3):37–51, Jun 1995.

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

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

[4] 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.

[5] E. Ferreira and B. Krogh. Switching controllers based on neural network estimates of stability regions and controller performance. Submitted to *Automatica Special Issue on Hybrid Systems* to appear Jan. 1999.

[6] E.D. Ferreira and B.H. Krogh. Controller scheduling by neural networks. In *Proc. 36th Conf. Decision Control*, San Diego, CA, Dec 1997.

[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 ii: Robustness. In *34th IEEE Conference on Decision and Control*, volume 2, pages 1750–5, New York, USA, Dec 1995.

[10] 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.

[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] Jean-Jacques. E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall Inc., 1991.

[13] C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

[14] 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.