REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1999	Final Progress Report 2/4/94-2/3/99

4. TITLE AND SUBTITLE

New Methods of Neural Network Training

5. FUNDING NUMBERS

DAAH04-94-G-0025

6. AUTHOR(S)

Robert F. Stengel

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Princeton University

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

ARO 30906.1-MA

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

New methods for training computational neural networks for dynamic system identification and control have been created, performance of the training algorithms has been analyzed, and the resulting neural networks have been evaluated. Computational neural networks are shown to have excellent potential for identifying the dynamic models of nonlinear systems and for controlling such systems over their entire operating space. Three topics were addressed:

- Aerodynamic model identification using sigmoid and radial-basis-function networks
- Control of the preferential oxidizer for a fuel-cell power system using a neural network
- Initializing a neural network (nonlinear) controller so that it replicates the characteristics of a gain-scheduled linear controller.

This research produced new training approaches that will allow future dynamic systems to work with higher accuracy, greater efficiency, and improved reliability.

14. SUBJECT TERMS	15. NUMBER OF PAGES
New methods for training computational neural networks for dynamic system identification and control.	14
	16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

Enclosure 1

MAE-3052

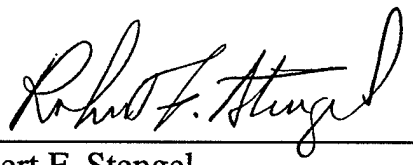
NEW METHODS OF NEURAL NETWORK TRAINING

Final Technical/Progress Report
US Army Research Office Grant No. DAAH04-94-G-0025
Princeton Account No. 150-6688

June 17, 1999

Prepared for:

U.S. ARMY RESEARCH OFFICE
Research Triangle Park, NC 27709



Robert F. Stengel
Professor, Principal Investigator
Princeton University
D-202 Engineering Quadrangle
Princeton, NJ 08544

Tel:	(609) 258-5103
FAX:	(609) 258-6109
E-mail:	stengel@princeton.edu
Internet URL:	http://www.princeton.edu/~stengel/

FOREWORD

New methods for training computational neural networks for dynamic system identification and control have been created, performance of the training algorithms has been analyzed, and the resulting neural networks have been evaluated. Computational neural networks are shown to have excellent potential for identifying the dynamic models of nonlinear systems and for controlling such systems over their entire operating space. Three topics were addressed:

- Aerodynamic model identification using sigmoid and radial-basis-function networks
- Control of the preferential oxidizer for a fuel-cell power system using a neural network
- Initializing a neural network (nonlinear) controller so that it replicates the characteristics of a gain-scheduled linear controller.

This research produced new training approaches that will allow future dynamic systems to work with higher accuracy, greater efficiency, and improved reliability.

TABLE OF CONTENTS

1. STATEMENT OF THE PROBLEM

Computational neural networks are motivated by input-output and learning properties of biological neural systems, though in mathematical application the network becomes an abstraction that may bear little resemblance to its biological model. Computational neural networks consist of *nodes* that simulate the *neurons* and *weighting factors* that simulate the *synapses* of a living nervous system. The nodes are nonlinear basis functions, and the weights contain knowledge of the system. Neural networks are good candidates for performing a variety of functions in *intelligent control systems* because they are potentially very fast (in parallel hardware implementation), they are intrinsically nonlinear, they can address problems of high dimension, and they can learn from experience. From the biological analogy, the neurons are modeled as switching functions that take just two discrete values; however, "switching" may be softened to "saturation," not only to facilitate learning of the synaptic weights but to admit the modeling of continuous, differentiable functions. Furthermore other nonlinear functions, such as radial basis functions and wavelets, can be used as activation functions.

The neural networks receiving most current attention are static expressions that perform one of two functions. The first is to *approximate multivariate functions* of the form,

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) \qquad (1$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are input and output vectors and $\boldsymbol{h}(\bullet)$ is the (possibly unknown) relationship between them. Neural networks can be viewed as *generalized spline functions* that identify efficient input-output mappings from observations. The second application is to *classify attributes*, much like decision trees.

An N-layer *feed-forward neural network* (FNN) represents the function by a sequence of operations,

$$\mathbf{r}^{(k)} = \mathbf{s}^{(k)}[\mathbf{W}^{(k-1)}\mathbf{r}^{(k-1)}] \triangleq \mathbf{s}^{(k)}[\eta^{(k)}] \text{ , } k = 1 \text{ to } N \qquad (2$$

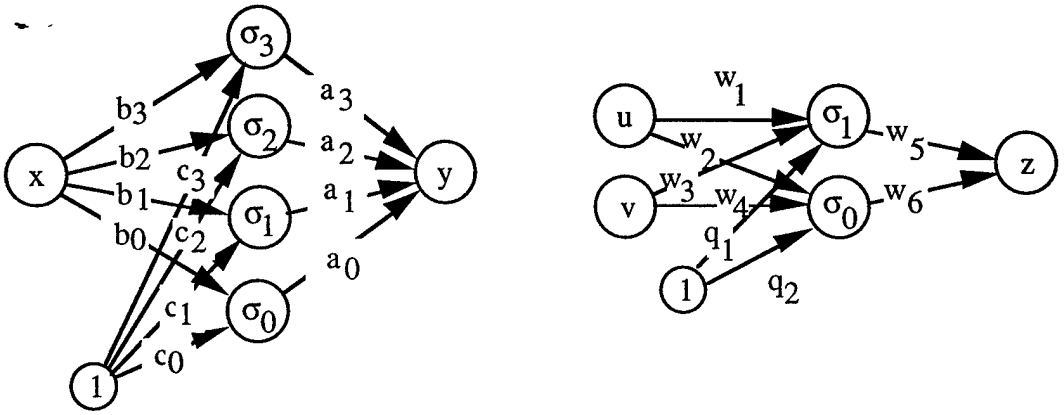
where $\boldsymbol{y} = \mathbf{r}^{(N)}$ and $\boldsymbol{x} = \mathbf{r}^{(0)}$. $\mathbf{W}^{(k-1)}$ is a matrix of weighting factors determined by the learning process, and $\mathbf{s}^{(k)}[\bullet]$ is an activation-function vector whose elements normally are identical, scalar, nonlinear functions $\sigma_i(\eta_i)$ appearing at each node:

$$\mathbf{s}^{(k)}[\eta^{(k)}] = [\sigma_1(\eta_1{}^{(k)}) \ldots \sigma_n(\eta_n{}^{(k)})]^T \qquad (3$$

One of the inputs to each layer may be a unity threshold element that adjusts the bias of the layer's output. Networks consisting solely of linear activation functions are of little interest, as they merely perform a linear transformation $\boldsymbol{H}$, thus limiting eq. 1 to the form, $\boldsymbol{y} = \boldsymbol{Hx}$. Figure 1 represents two simple feed-forward neural networks. Each circle represents an arbitrary, scalar, nonlinear function $\sigma_i(\bullet)$ operating on the sum of its inputs, and each arrow transmits a signal from the previous node, multiplied by a weighting factor.

More than one set of weights could produce the same functional relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$. Training sessions starting at different points could produce different sets of weights that yield identical outputs. The result presages a well-known problem of network weight determination: multiple local minima in error-minimizing solutions that may

prevent the identification of the best network for representing a given function. The unstructured feed-forward network may not have compact support (i.e., its weights may have global effects) if its basis functions do not vanish for large magnitudes of their arguments.



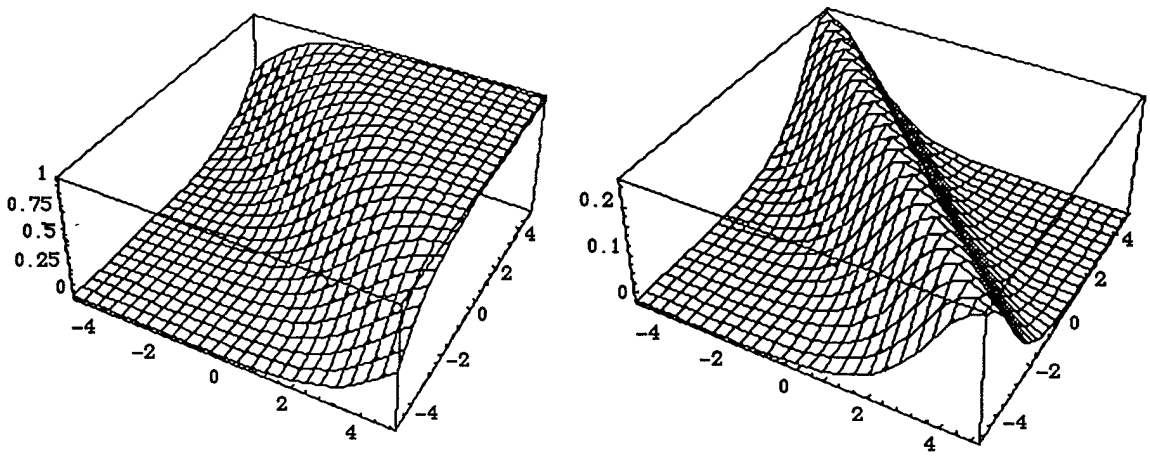
a) Single-Input/Single-Output Network. b) Double-Input/Single-Output Network.

Figure 1. Two Feed-forward Neural Networks.

The *sigmoid* is commonly used as the artificial neuron. It is a saturating function defined variously as $\sigma(\eta) = 1/(1 + e^{-\eta})$ for output in (0,1) or $\sigma(\eta) = (1 - e^{-2\eta})/(1 + e^{-2\eta}) = \tanh \eta$ for output in (-1,1). Recent results indicate that any continuous mapping can be approximated arbitrarily closely with sigmoidal networks containing a single hidden layer (N = 2). Symmetric activation functions like the *Gaussian radial basis function* ($\sigma(\eta) = e^{-\mathbf{r}^T \mathbf{W} \mathbf{r}}$) have better convergence properties for many functions and have more compact support as a consequence of near-orthogonality. In such case, eq. 2 is rewritten as

$$\mathbf{r}^{(k)} = \mathbf{s}^{(k)}[\mathbf{r}^{(k-1)^T} \mathbf{W}^{(k-1)} \mathbf{r}^{(k-1)}] \triangleq \mathbf{s}^{(k)}[\eta^{(k)}] \text{ , } k = 1 \text{ to } N \qquad (4$$

In control application, neural networks perform functions analogous to gain scheduling or nonlinear control. Consider the simple two-input network of Fig. 1b. The scalar output and derivative of a single sigmoid with unit weights are shown in Fig. 2. If u is a fast variable and v is a slow variable, choosing the proper weights on the inputs and threshold can produce a gain schedule that is approximately linear in one region and nonlinear (with an inflection point) in another. More complex surfaces can be generated by increasing the number of sigmoids. If u and v are both fast variables, then the sigmoid can represent a generalization of their nonlinear interaction. In that regard, the FNN may represent a nonlinear control function directly, or it may represent a nonlinear dynamic model that is to be inverted by separate control logic.



a) Sigmoid. b) x-Derivative of Sigmoid.

Figure 2. Example of Sigmoid Output with Two Inputs.

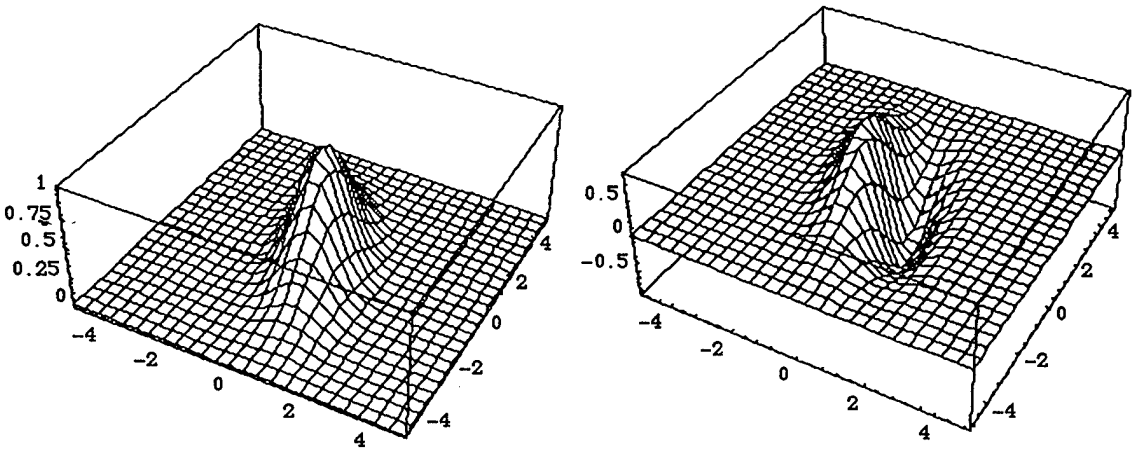
For comparison, a typical radial basis function produces the output shown in Fig. 3. Whereas the sigmoid has a preferred input axis and simple curvature, the RBF admits more complex curvature of the output surface, and its effect is more localized. The most efficient nodal activation function depends on the general shape of the surface to be approximated, as well as on the importance of compact support.

The cerebellar model articulation controller (CMAC) is an alternate network formulation with somewhat different properties but similar promise for application in control systems. The CMAC performs table look-up of a nonlinear function over a particular region of function space. CMAC operation can be split into two mappings. The first maps each input into an *association space A*. The mapping generates a *selector vector* $\boldsymbol{a}$ from overlapping *receptive regions* for the input. The second mapping, R, goes from the selector vector $\boldsymbol{a}$ to the scalar output y through the weight vector $\boldsymbol{w}$, which is derived from training:

$$y = \boldsymbol{w}^T \boldsymbol{a} \qquad (5$$

Training is inherently local, as the extent of the receptive regions is fixed. The CMAC has quantized output, producing "staircased" rather than continuous output.

The FNN and CMAC are both examples of *instantaneous networks*, that is, their outputs are essentially instantaneous: given an input, the speed of output depends only on the speed of the computer. *Dynamic networks* rely on stable resonance of the network about an equilibrium condition to relate a fixed set of initial conditions to a steady-state output. Bidirectional Associative Memory (BAM) networks are nonlinear dynamical systems that subsume Hopfield networks, Adaptive-Resonance-Theory (ART) networks, and Kohonen networks. Like FNN, they use binary or sigmoidal neurons and store knowledge in the weights that connect them; however, the "neural circuits" take time to stabilize on an output. While dynamic networks may operate more like biological neurons, which have a *refractory period* between differing outputs, computational delay degrades control functions; hence, instantaneous networks are preferred in many identification and control applications.



a) Radial Basis Function (RBF). b) x-Derivative of RBF.

Figure 3. Example of Radial Basis Function Output with Two Inputs.

1.1 Aerodynamic Model Identification

Aerodynamic model identification is an example of the generic multivariate function approximation problem with certain constraints on the training space. The multivariate functions -- in this case, aerodynamic coefficients that have nonlinear dependence on several variables -- are embedded in high-order ordinary differential equations. The goal is to derive the aerodynamic models from measurements of physically realizable trajectories in the state and control space in two steps: generation of the training set and training of the network. First, an extended Kalman filter (EKF) processes simulated or actual measurements to minimize the effects of measurement error, to account for likely disturbance effects on the dynamical system, and to estimate the forces and moments related to the aerodynamic coefficients. The state, control, forces, and moments are the training set for a feed-forward neural network (FNN) (also called a "back-propagation network," although different training algorithms are used here). Second, the FNN, containing a single hidden layer of sigmoid or radial-basis-function "neurons," is trained on this set using a separate EKF, a genetic algorithm (GA), or a combination of the two.

1.2 Cerebellar Model Articulation Controller

The *cerebellar model articulation controller* (CMAC) is based on a sequence of memory and data mappings rather than on interconnected neurons. The CMAC maps an input vector to an output vector in two steps. In the first mapping, the input is discretized and quantized in the receptive region; in the second mapping, outputs of the receptive region form the conceptual or association memory that produces the desired output. There also may be a hashing region that compresses memory requirements with little degradation in performance. For the application described in Section 2.2, the CMAC operates in parallel with a proportional-integral-derivative (PID) controller, which provides initial stabilization as the CMAC learns its weights through error-gradient-based training. The CMAC gradually takes over from the PID controller, providing improved nonlinear control in the process.

1.3 Classical/Neural Synthesis of Nonlinear Control Systems

Classical/neural synthesis of control systems combines the most effective elements of old and new design concepts to produce better control systems. There is considerable precedent for applying gain-scheduled linear controllers to nonlinear systems, especially those that can be approximated as linear-parameter-varying systems; however, a means for transferring the insights gained from these linear controllers to nonlinear controllers remains to be identified. This research initiated a new approach for designing nonlinear control systems that takes advantage of prior knowledge and experience in designing linear controllers, while capitalizing on the broader capabilities of adaptive, nonlinear control theory and artificial (or computational) neural networks. Central to this new approach is the recognition that the gradients of a nonlinear control law must represent the gain matrices of an equivalent, locally linearized controller. Hence, a family of satisfactory linear controllers specified over the operating envelope of the system forms a suitable starting point for the definition of a global nonlinear controller. The initial specification for the controller, which can be represented by neural networks, retains the stability, performance, and robustness guarantees of the linearized model for small perturbations. On-line learning improves control response for large, coupled motions, accounting for differences between actual and assumed dynamic models and for nonlinear effects not captured in the linearization.

2. SUMMARY OF IMPORTANT RESULTS

2.1 Aerodynamic Model Identification

Model identification results focus on the speed and accuracy of FNN training, the effects of using alternate nodal activation functions, the ability of the network to generalize from trajectory training data, and the ability of a pre-trained network to learn a new, localized feature of the function. Six methods of training a sigmoidal FNN for model identification are compared in [3]. (Related research on the use of a GA for learning the design parameters of a linear compensator is documented in [10].) The task is to estimate the lift coefficient of an aircraft as a function of up to three variables. Of the six methods, (four EKFs, a genetic algorithm, and a hybrid GA-EKF), the most successful is an extended Kalman filter with fictitious process noise. It had been anticipated that the hybrid method, which combines an initial global search by GA with the strong convergence properties of the EKF, would prove efficient, and it was four to nine times faster than either GA or EKF alone. However, the EKF with process noise proved quickest to converge, an additional five to 20 times faster than the GA-EKF.

An apparent advantage of using a radial basis function (RBF) for a nodal activation function rather than a sigmoid is that its domain of significant effect is more compact [4,8]. The sigmoid extends from zero to one across the input domain, while the RBF produces an exponential "bump" at its center, trailing off to zero elsewhere. If features of the training function change locally, it seems likely that an RBF network would learn these features more quickly than a sigmoid network and with less disturbance to the trained function elsewhere. Our investigation confirms this hypothesis. The RBF networks typically converge to the new feature several times faster than the sigmoid networks.

Nevertheless, RBF networks, as typically defined, suffer from a "curse of dimensionality," growing exponentially and inefficiently as the number of independent variables increases. We have defined a new, hierarchical type of RBF network that is espe-

cially well-suited to on-line learning [9]. The model update is developed over time through the use of three approximations, all housed within one neural network. Gaussian radial basis functions with the centers placed on grids of different resolutions serve as the network activation functions, or nodes, where each grid is assigned a set of width ranges. Network training is reduced to a nodal selection/output-weight calculation problem. The network considers two surfaces: the Baseline and Residual-Surface. The Baseline portion of the network is initialized prior to flight. Its parameters remain fixed during system operation while the residual surface, that surface defined by the difference between actual flight data and the current Baseline Approximation, is captured by the Residual-Surface Approximation. This approximation is also built with two surfaces: the Interim and Flight Approximations. The Flight model is generated during system operation using a selection procedure based on nodal output magnitudes and the Givens least-squares algorithm for output-weight calculation. The Interim Approximation is generated at fixed time intervals, and it efficiently replaces the Flight model. The information held in the Interim model is used to update the Baseline Approximation. The Baseline and Interim surfaces are generated with a new, fast, accurate training procedure for problems that include several inputs. Models must be able to generalize or respond accurately to data not seen during training; this quality is monitored during all phases of learning and considered for approximation establishment.

2.2 Cerebellar Model Articulation Controller

Ground vehicles fueled by hydrocarbons or alcohols and powered by proton exchange membrane (PEM) fuel cells address world air quality and fuel supply concerns while avoiding hydrogen infrastructure and on-board storage problems. Instead of carrying gaseous hydrogen on vehicles, fuel cell developers are exploring on-board fuel processors that convert a hydrogen-containing fuel – such as methanol, ethanol, or gasoline – into a hydrogen-rich gas. These fuels are easier to store and distribute than hydrogen gas, and fuels such as gasoline have a production and distribution infrastructure already in place. However, a major concern when operating PEM fuel cells on the hydrogen-rich gas from a fuel processor is the poisoning of the fuel cell's anode catalyst, and thus the degradation of vehicle performance, by carbon monoxide. The gas stream or "reformate" from a fuel processor contains hydrogen, carbon dioxide, water, and carbon monoxide. Care must be taken to reduce the carbon monoxide level in the gas to only a few parts per million before it enters the fuel cell. In an on-board fuel processor, the final carbon monoxide clean-up step is performed by a relatively new catalytic reactor called the preferential oxidizer or PrOx. Reduction of the carbon monoxide concentration in the on-board fuel processor's hydrogen-rich gas by the preferential oxidizer (PrOx) under dynamic conditions is crucial to avoid poisoning of the PEM fuel cell's anode catalyst and thus malfunction of the fuel cell vehicle.

The CMAC has been used as a nonlinear controller for a fuel cell's PrOx [5,6]. The gas flow rate and temperature of the PrOx have nonlinear effects on the reaction and must be carefully controlled to maximize the performance and life of the fuel cell -- hence the need for an adaptive, nonlinear controller. The CMAC is shown to perform better than a PID controller alone, given slow and inaccurate sensors, rapid fuel processor transients, and systematic fuel processor changes due to aging.

A dynamic control scheme for a single-stage, tubular, cooled PrOx was been shown to perform better than conventional industrial controllers. The hybrid control system contains a CMAC artificial neural network in parallel with a conventional PID controller. A computer simulation of the preferential oxidation reactor illustrated the abilities of the controller and compared its performance to the performance of conventional controllers. Realistic input patterns were generated for the PrOx by using models

of vehicle power demand and upstream fuel processor components to convert the speed sequences in the Federal Urban Driving Schedule (FUDS) to PrOx inlet temperatures, concentrations, and flow rates. The hybrid controller generalizes well to novel driving sequences after being trained on other driving sequences with similar or slower transients. Although it is similar to the PID in terms of software requirements and design effort, the hybrid controller performs significantly better than the PID in terms of hydrogen conversion setpoint regulation and PrOx outlet carbon monoxide reduction.

2.3 Classical/Neural Synthesis of Nonlinear Control Systems

We consider dynamic systems described by the nonlinear ordinary differential equation:

$$\dot{x} = f\left[x(t), p(t), u(t), w(t)\right] \qquad (6$$

x is the $(n \times 1)$ plant state, p is a $(\ell \times 1)$ vector of plant and observation parameters, u is the $(m \times 1)$ control, and w is a $(s \times 1)$ vector of disturbance effects. The equation may represent a "lumped-parameter" system, or it may be an approximation to an unsteady partial differential equation. Plant motions, controls, and disturbances are sensed in the $(r \times 1)$ output y_S,

$$y_S(t) = h_S\left[x(t), p(t), u(t), w(t)\right] \qquad (7$$

and the measurement, z, is subject to uncertainty, n:

$$z(t) = y_S(t) + n(t) \qquad (8$$

The design objective is to specify a control law of the general form

$$u(t) = c\left[z(t), p(t), y_C(t)\right] \qquad (9$$

that has two properties: it achieves mission goals, as expressed by the $(r_C \times 1)$ command input, $y_C(t)$, and it furnishes adequate stability and transient response, assuring that excursions from $y_C(t)$ caused by disturbance or measurement error are acceptably small and do not require excessive control use.

The command input, $y_C(t)$, can be viewed as some desirable (possibly nonlinear) combination of state and control elements, and its dimension is less than or equal to the number of independent controls ($r_C \leq m$):

$$y_C(t) = h_C\left[x(t), u(t)\right] \qquad (10$$

It could be the result of external trajectory planning (e.g., following a prescribed, possibly optimal, path), or it may be due to a loosely defined, subjective process (e.g., the expression of a human operator's intent).

For the discussion, we address the more limited goal of control with perfect measurements, simplifying the control law to

$$u(t) = c\left[x(t), p(t), y_C(t)\right] \tag{11}$$

$c\left[x(t), p(t), y_C(t)\right]$ may be a functional, containing functions of its arguments, such as integrals or derivatives of $x(t)$ and $y_C(t)$. We always can write the control law as the sum of a nominal effect and a perturbed effect

$$\begin{aligned} u(t) &= c_o\left[x_o(t), p(t), y_{C_o}(t)\right] + \Delta c\left[x(t), p(t), y_C(t)\right] \\ &= u_o(t) + \Delta u(t) \end{aligned} \tag{12}$$

where, for simplicity, we assume that $p(t)$ is known without error. Anticipated values of the state and command are x_o and y_{C_o}, and actual values are x and y_C where

$$\begin{aligned} x &= x_o + \Delta x \\ y &= y_o + \Delta y \end{aligned} \tag{13,14}$$

Hence, the control law can be expressed as

$$u(t) = c_o\left[x_o(t), p(t), y_{C_o}(t)\right] + \Delta c\left[x_o(t), p(t), y_{C_o}(t), \Delta x(t), \Delta y_C(t)\right] \tag{15}$$

For sufficiently small state and command perturbations, the perturbed control effect is linear in Δx and Δy_C and it can be written as

$$\begin{aligned} \Delta u(t) = \Delta c[\bullet] &= \frac{\partial c}{\partial x}\left[x_o(t), p(t), y_{C_o}(t)\right]\Delta x + \frac{\partial c}{\partial y_C}\left[x_o(t), p(t), y_{C_o}(t)\right]\Delta y_C \\ &= C_x \Delta x + C_y \Delta y_C \end{aligned} \tag{16}$$

C_x and C_y contain the m gradients of the control law evaluated at $\left[x_o(t), p(t), y_{C_o}(t)\right]$. Equation 16 can be viewed as a linear, gain-scheduled control law which, when combined with $c[\bullet]$, provides a close approximation to the exact nonlinear controller (eq. 11) for small Δx and Δy_C:

$$u(t) = c_o\left[x_o(t), p(t), y_{C_o}(t)\right] + C_x \Delta x + C_y \Delta y_C \tag{17}$$

It is clear that knowledge of $c\left[x(t), p(t), y_C(t)\right]$ at a single point and of C_x and C_y over the operating range of $\left[x(t), p(t), y_C(t)\right]$ (or some suitably dense set in the space) is equivalent to knowledge of $c\left[x(t), p(t), y_C(t)\right]$ over the same range. Put another way, given a nonlinear control in the form of eq. 9, the corresponding gain-scheduled control law is readily found. Our objective is to find efficient ways of solving the inverse problem, that is, to derive a nonlinear control law from a satisfactory gain-scheduled control law.

Gain-scheduled control laws are based on a set of linear, time-invariant (LTI) control laws specified throughout the plant's operating region. Given the dynamic system of eq. 6, a first-degree expansion can be written:

$$\begin{aligned}\dot{x}(t) &= \dot{x}_o(t) + \Delta\dot{x}(t) \\ &= f\left[x_o(t), p(t), u_o(t), w_o(t)\right] + \Delta f\left[x_o(t), p(t), u_o(t), w_o(t), \Delta x(t), \Delta u(t), \Delta w(t)\right] \\ &\approx f\left[x_o(t), p(t), u_o(t), w_o(t)\right] + \frac{\partial f}{\partial x}\Delta x(t) + \frac{\partial f}{\partial u}\Delta u(t) + \frac{\partial f}{\partial w}\Delta w(t) \\ &= f[\bullet] + F\Delta x(t) + G\Delta u(t) + L\Delta w(t)\end{aligned} \tag{18}$$

The Jacobian matrices, F, G, and L, are evaluated at selected operating points, and the perturbation model is:

$$\begin{aligned}\Delta\dot{x}(t) = {} & F\left[x_o(t), p(t), u_o(t), w_o(t)\right]\Delta x(t) \\ & + G\left[x_o(t), p(t), u_o(t), w_o(t)\right]\Delta u(t) + L\left[x_o(t), p(t), u_o(t), w_o(t)\right]\Delta w(t)\end{aligned} \tag{19}$$

This model is almost a linear, parameter-varying (LPV) plant, "almost" because the system matrices depend on $x_o(t)$, as well as the remaining variables. In most applications, effects of parameter variation are ignored because time-varying dynamic effects are small, and [F, G, L] is treated as a set of LTI plant models. Linear control gains (e.g., C_x and C_y) are computed for each LTI model, and the control law is implemented with interpolation of gains to intermediate operating points. In past applications, the number of interpolating variables has been kept small.

Future research will identify a means of greatly expanding the number of (independent) interpolating variables, affording an improvement in comparison to gain-scheduled controllers. More important, it will provide an excellent initialization point for the neural-network controller [7]. Given C_x, C_y, and the corresponding equilibrium points at each operating point, the corresponding nonlinear control law (eq. 11) will be generated.

We assume that the LTI control laws used for this pre-training phase satisfy accepted engineering design criteria, based on design principles described in our earlier work. For example, we have shown how to use Monte Carlo evaluation and genetic algorithms to design robust linear, quadratic-Gaussian (LQG) controllers that satisfy classical design criteria. That process begins with conventional stability and performance specifications (e.g., negative eigenvalues, suitable limits on rise time, settling time, and control usage), and it generates desired values of C_x, C_y, and the equilibrium points. In the process, the corresponding weighting matrices for quadratic cost functions, such as,

$$J = \lim_{t_f \to \infty} \frac{1}{2}\int_0^{t_f} L\left[x(\tau), u(\tau)\right]d\tau = \lim_{t_f \to \infty} \frac{1}{2}\int_0^{t_f}\left[x^T(\tau)Qx(\tau) + 2x^T(\tau)Mu(\tau) + u^T(\tau)Ru(\tau)\right]d\tau \tag{20}$$

are found. These cost functions become critical elements for on-line learning.

On-line training of a neural network is based on the minimization of an error function; the error function chosen here takes the form of eq. 15. The result is a dynamic programming problem, in which the nonlinear control law minimizes the expected value of a cost function such as eq. 20. We replace the cost function, J, by the value function, V, which is defined as

$$V(t) = -\lim_{t_f \to \infty} \frac{1}{2} \int_{t_f}^{t} L[x(\tau), u(\tau)] d\tau \tag{21}$$

and seek to minimize the value function via the Hamilton-Jacobi-Bellman (HJB) equation by choice of control:

$$\frac{\partial V^*}{\partial \tau}[x^*(t), t] = -\min_u \left[L[x^*(t), u(t), t] + \frac{\partial V^*}{\partial x}[x^*(t), t] f[x^*(t), u(t), t] \right] \tag{22}$$

Because the problem is stochastic, the value function is recast as the expected value of the integral in eq. 16:

$$V(t) = -E\left[\lim_{t_f \to \infty} \frac{1}{2} \int_{t_f}^{t} L[x(\tau), u(\tau)] d\tau \right] \tag{23}$$

Hence, the elements of eq. 17 are expectations rather than deterministic functions. With sufficient smoothness, the corresponding control, $u^*(\tau)$, along the optimal trajectory, $x^*(\tau)$, is specified by the optimality condition

$$0 = \frac{\partial}{\partial u} \left[L[x^*(t), u(t), t] + \frac{\partial V^*}{\partial x}[x^*(t), t] f[x^*(t), u(t), t] \right] \tag{24}$$

While this condition implicitly specifies the control, our goal is to derive an explicit relation in the form of eq. 11. Solution of this problem is afforded by the *adaptive critic architecture*. This architecture consists of an *action network*, which expresses the control law (eq. 11), plus a *critic network*, which estimates the $\partial V^*/\partial x$ required in eq. 24. For the linear case, optimal control perturbations can be calculated as

$$\Delta u(t) = -R^{-1} G^T \frac{\partial V^*}{\partial x} = -R^{-1} G^T P(t) \Delta x(t) = C_x \Delta x(t) \tag{25}$$

where $P(t)$ is the solution to the well-known matrix Riccati equation. Hence, there is an intimate relationship between $\partial V^*/\partial x$ and the gradient of the control surface C_x.

3. LIST OF PUBLICATIONS AND PRESENTATIONS

1. R. F. Stengel, *Autonomous Control Systems for Space Flight*, NASA/DoD Automation/Operations Workshop, July 1995, Arlington, VA.

2. R. F. Stengel, *New Methods for Neural Network Training*, 1996 ARO Annual Meeting of Investigators in Systems and Control and Software Knowledge-Based Systems, February 1996, Research Triangle Park.

3. K. Richardson and R. F. Stengel, "A Comparison of Neural Network Training Algorithms," *Proc. 1997 AIAA Atmospheric Flight Mechanics Conference*, AIAA-97-3713, New Orleans, Aug. 1997, pp. 598-604.

4. K. Richardson and R. F. Stengel, "Effects of Localized Inputs on Neural Network Training," *Proc. 1998 AIAA Atmospheric Flight Mech. Conf.*, Boston, Aug. 1998.

5. L. C. Iwan, "Design of a CMAC Neural Network Controller for the Preferential Oxidation Reactor in a Fuel Cell Vehicle's On-Board Methanol Fuel Processor," MAE MSE Thesis 2098-T, Princeton University, Princeton, NJ, Apr. 1997.

6. L. Iwan and R. F. Stengel, "The Application of Neural Networks to Fuel Processors for Fuel Cells," *Proc. 37th IEEE Conf. Dec. and Cont.*, Tampa, Dec. 1998, pp. 1585-1590.

7. S. Ferrari and R. F. Stengel, "Proportional-Integral Neural Network Controller," to be presented at 7th International Workshop on Computer Aided Systems Theory and Technology, Vienna, Sept. 1999.

8. K. Richardson and R. F. Stengel, "Aerodynamic Coefficient Modeling for Nonlinear Control," submitted to the 22nd Congress of the International Council of the Aeronautical Sciences, Harrowgate, UK, Aug. 2000.

9. K. Richardson, "Identification of Aerodynamic Coefficients with Neural Networks," Ph. D. Thesis, Princeton University, Princeton, NJ (in preparation).

9. C. I. Marrison, and R. F. Stengel, "Robust Control System Design Using Random Search and Genetic Algorithms," *IEEE Trans. Automatic Control*, Vol. 42, No. 6, June 1997, pp. 835-839.

4. LIST OF PARTICIPATING SCIENTIFIC PERSONNEL

Robert F. Stengel	Principal Investigator
Laura Iwan	M. S. E. student. Degree completed: Dec. 1997.
Kristina Richardson	Ph. D. student. Expected date of completion: Aug. 1999.
Silvia Ferrari	Ph. D. student. Expected date of completion: Aug. 2001.

5. REPORT OF INVENTIONS

N/A