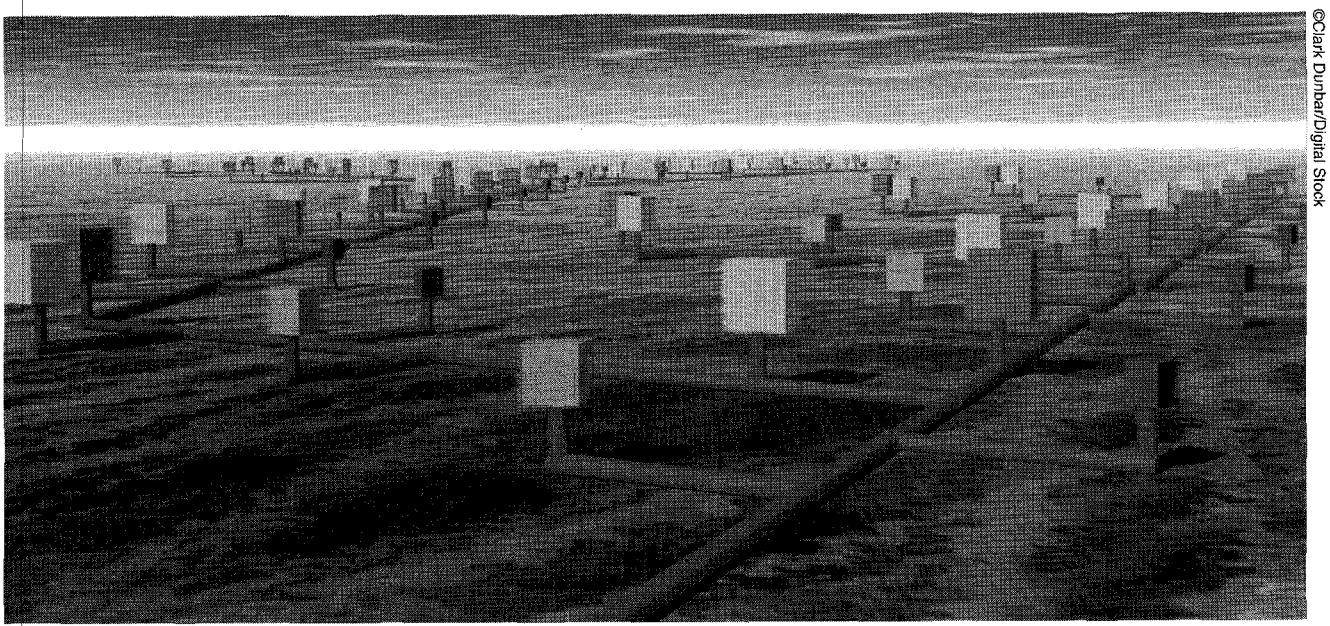
A Systematic Classification of Neural-Network-Based Control

Mukul Agarwal



Successful industrial applications and favorable comparisons with conventional alternatives have motivated the development of a large number of schemes for neural-network-based control. Each scheme is usually composed of several independent functional features, which makes it difficult to identify precisely what is new in the scheme. Help from available overviews is therefore often inadequate, since they usually discuss only the most important overall schemes. This work breaks the available schemes down to their essential functional features and organizes the latter into a multi-level classification. The classification reveals that similar schemes often get placed in different categories, fundamentally different features often get lumped into a single category, and proposed new schemes are often merely permutations and combinations of the well-established fundamental features.

A preliminary version of this article was presented at the 3rd IEEE Conference on Control Applications in Glasgow, U.K., in August 1994. *The author is with Systems Engineering Group, TCL, Eidgenossische Technische Hochschule, CH-8092 Zurich, Switzerland. Tel.: +41-1-632-3113. Fax: +41-1-632-1222. Email: agarwal@tech.chem.ethz.ch.*

Introduction

Neural networks have found growing success in diverse industrial control applications [1]. In rigorous comparisons neural-network controllers have fared better than well-established conventional options when the plant characteristics are poorly known [2-6]. Fast parallel computation and use of generic function forms for complex nonlinear mappings have motivated the proposal of a growing number of control schemes involving neural networks. Given the multitude of themes and variations, it is difficult to isolate what is new in any proposed scheme. Little guidance in this respect is provided by the few available overview papers, which unfortunately do not develop a structured classification of the control schemes. Thus Werbos [7] suggests a linear division into five strategies, whereas Hunt *et al.* [8] distinguish between nine different strategies. Others [9-15] report subsets of this list, the last adding one new entry to it. These overviews offer no justification for the used division, no relationships between the divided classes, and no subdivision within a class, leaving a seemingly complex maze of several two-valued dimensions, e.g., supervised/unsupervised learning, direct/indirect inverse, generalized/specialized learning.

This work develops a comprehensive systematic classification of the neural-network-based control schemes proposed in

 0272-1708/97/$10.00©1997IEEE

the literature. A precise multi-level structure results, which isolates and classifies without spurious overlaps the distinct functional features that appear in different permutations and combinations in the various proposed schemes. The classification is useful for assessing the novelty of any given scheme and guiding the development of further schemes. The scope of the work is limited to taxonomy; coverage of the literature and discussion of relative merits of the schemes are provided only to the extent necessary to motivate and develop the classification. Issues such as theoretical analyses, numerical and mathematical properties, closed-loop stability, convergence of training, strengths and limitations of the neural approach, comparisons with non-neural controllers, and directions for future research are left to surveys and review papers, some of which are cited

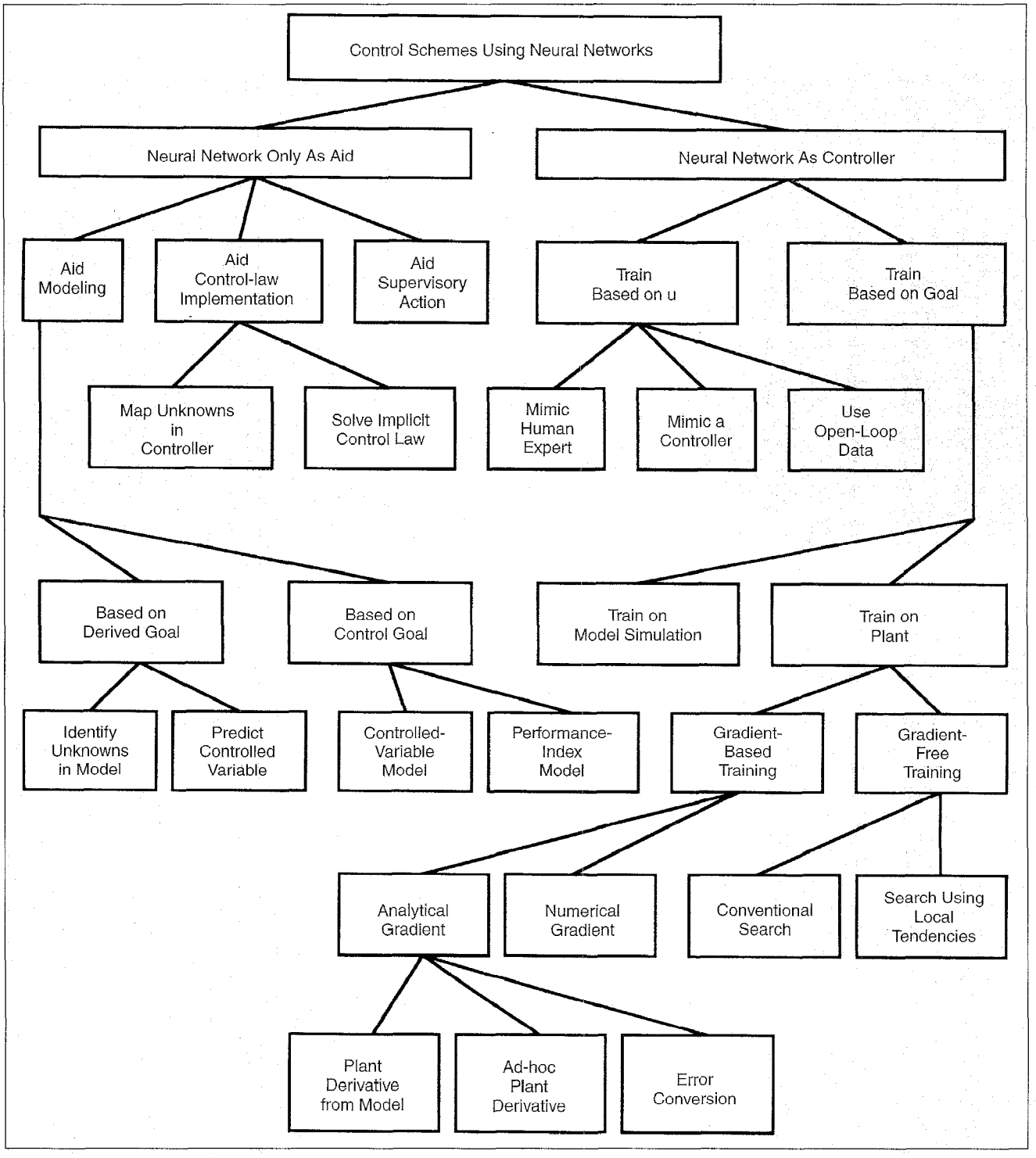


Fig. 1. Overview of the classification.

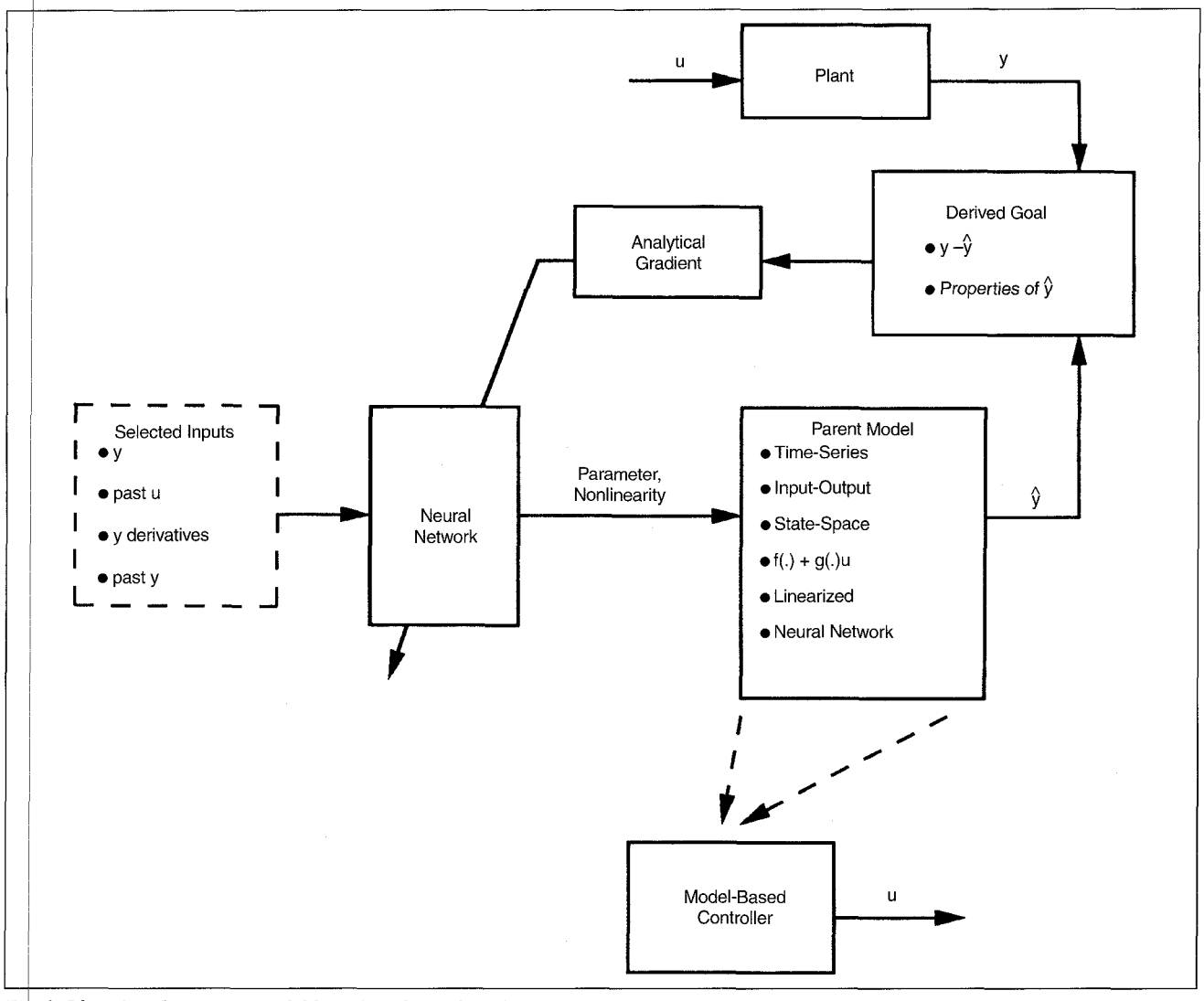


Fig. 2. Identify unknowns in model based on derived goal.

above. The choice of the cited literature does not reflect any (necessarily subjective) judgment about the quality or the importance of any article that happens to have been included or not included. The choice has been guided by completely value-free criteria, such as economizing on the total number of references by preferring those that illustrate more than one concept, or enabling easier (backward) search of the uncited literature by preferring newer articles to older ones, or ensuring better availability by preferring journal articles to papers in conference proceedings.

Nomenclature such as supervised learning, reinforcement learning, adaptive neural control, etc., that has been used in the literature ambiguously to denote fundamentally different schemes [16] is deliberately avoided. Ambiguous neural-network terminology, e.g., hidden nodes, recurrent networks, etc., is replaced by simple descriptive phrasing. In order to illustrate the unavoidable basic terminology for the unfamiliar reader, a neural network can be regarded simply as a generic black-box function that maps certain signals called network inputs to signals called network outputs. The black box comprises processing units called nodes or neurons, each of which outputs a signal that is a generic nonlinear function of the sum of the signals received by the unit. Sources of received signals are specified through connections; nodes receiving all signals from identical sources (except any signals from each other or from themselves) comprise a group of nodes called a layer. Each signal received by a node is preprocessed through a multiplicative weighting parameter called a weight. Identification of the network weights so as to optimize a criterion dependent on the network outputs is called training or learning of the network. The data used for this identification are called the training data/set and those used for subsequent validation are called the test data/set; each data entry in either set is referred to as a pattern. Training is based on an analytical gradient when the sensitivity of each function appearing in the path that maps the network outputs to the training criterion is known. In a procedure, called backpropagation, for obtaining the analytical gradient, the local sensitivity of the training criterion, e.g., the error whose square is to be minimized, is backpropagated through this path to determine the sensitivity with respect to the network weights. When the training data contain values of the network outputs, the trained network is said to store these outputs; in application, the network is said to retrieve or interpolate its outputs for any given

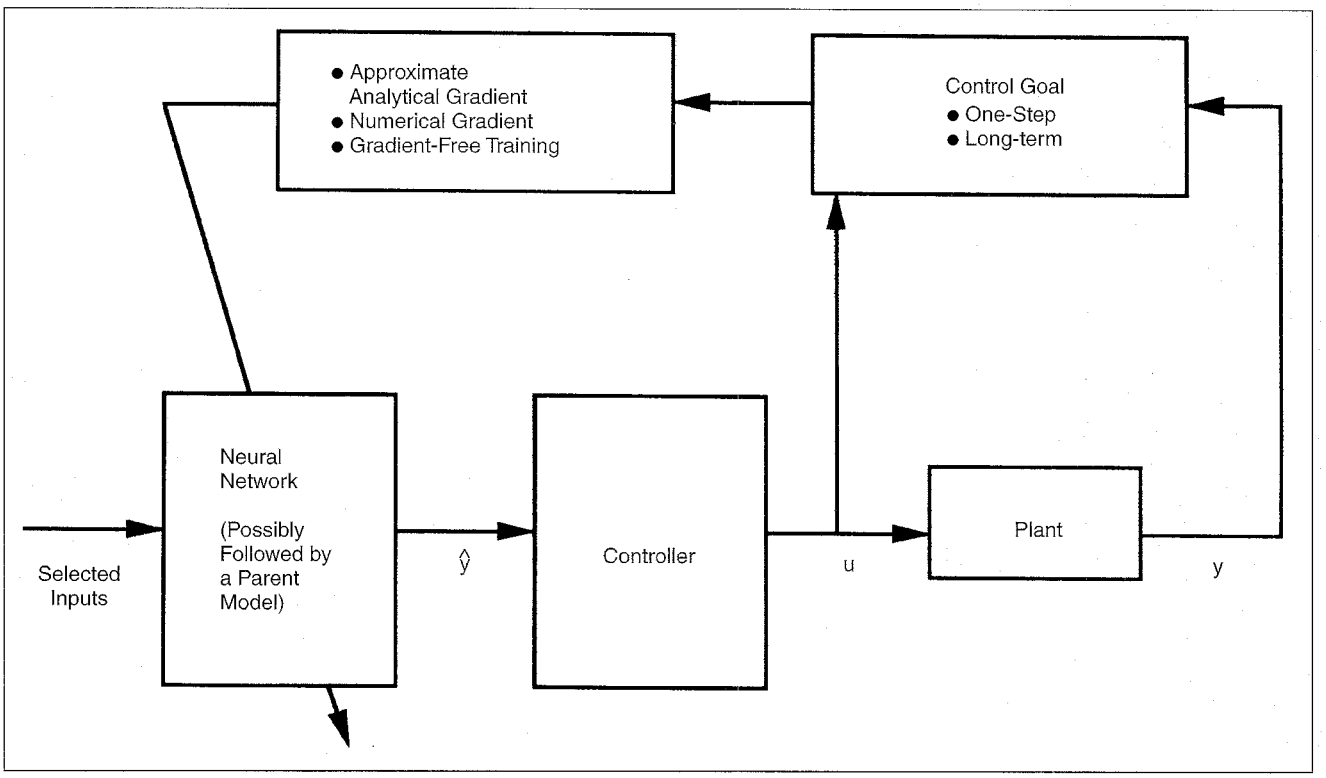


Fig. 3. Controlled-variable model based on control goal.

network-input value. In addition to the so-called feature-extraction task of identifying a functional mapping, the structure of neural networks allows them to be employed also for classification and optimization tasks.

An overview of the proposed classification is shown in Fig. 1. The relatively limited option of using neural networks to merely aid a non-neural controller is further classified in the following section. Of the schemes in which the controller itself is a neural network, the section "Train Based on u" classifies the alternative where control-input signals u are available for training the neural controller and the section "Train Based on Goal"classifies the option where the network devises the needed control strategy on its own, based on the ultimate control objective. Certain general enhancement features, each relevant to many branches in the classification tree, are deferred to the penultimate section.

Neural Network Only as Aid

As efficient nonlinear function approximators, pattern classifiers, and optimizers, neural networks are used to relieve complex diagnostics, tedious modeling, unwieldy tuning, or excessive computational effort in conventional control schemes. Three distinct roles of neural network as an aid are possible. First, the network assists or replaces a conventional model that is based either on a derived goal (first subsection), such as minimum prediction error of the controlled variable, or directly on the control goal (second subsection), such as minimization of a performance index. Second, the network simplifies implementation of a control law by storing certain controller parameters or by solving an implicit control law (third subsection). Third, the network performs diagnostics and recommends supervisory actions for a lower-level control system (fourth subsection).

Aid Modeling Based on Derived Goal

The derived modeling goal of minimizing the model prediction error is sub-optimal with respect to the overall control goal, but allows flexible use of the model for different control goals and easier training of the network weights due to absence of the unknown plant in the sensitivity path that is relevant for obtaining an analytical gradient.

Identify Unknowns in Model. In this scheme, the network supplies parameters or nonlinear functions to a parent model that yields predictions $\hat{y}$ of the controlled variable y; the prediction error is backpropagated to train the network (Fig. 2). On-line updates of the model parameters and nonlinearities provided by the network then enable indirect adaptive model-based control. Parameters of time-series [17], input-output [18], and state-space innovations [19, 20] models, general input nonlinearities [21], output nonlinearities [22], as well as nonlinearities f and g in the affine model $f(.) + g(.)u$ used in linearizing feedback control [23-25] have commonly been identified in this way. The on-line interpolation may be more reliable when a separate network is used for each variable in order to avoid coupling effects [26].

Similarly, networks have been used to identify linearization transformations involved in linearizing feedback control. Nonlinear functions involved in the linearization transformations have been identified as networks connected to a parent model (see Fig. 2) that itself is a neural network [27, 28], or as networks trained based on prediction errors in the controlled-variable time-derivatives [29]. Networks directly representing the linearization transformations embedded in a linearized model have been trained to achieve prescribed properties of the linearized-model output [30].

Predict Controlled Variable. In this scheme, the neural network identifies the entire model predicting the controlled vari-

able and is trained to minimize its prediction error. The network yields predictions of the measured output y over a time horizon, either directly using multiple network outputs, or recursively using a single network output [31]. Use of a separate network to predict the output for each future time interval is advisable, in order to avoid confounding coupling effects [32]. When an on-line unmeasured variable x, usually the process state, is controlled, the network models an estimate [33-35] or recursive predictions $\hat{x}$ of x, and is trained using off-line measurements of x. Using feedback networks (see the section "Feedback Network Structure") based on state-space representations may yield a better mapping for both y-control [36] and x-control [37].

This model is useful in controllers that do not require an analytical model, but only predictions of the controlled variable. The

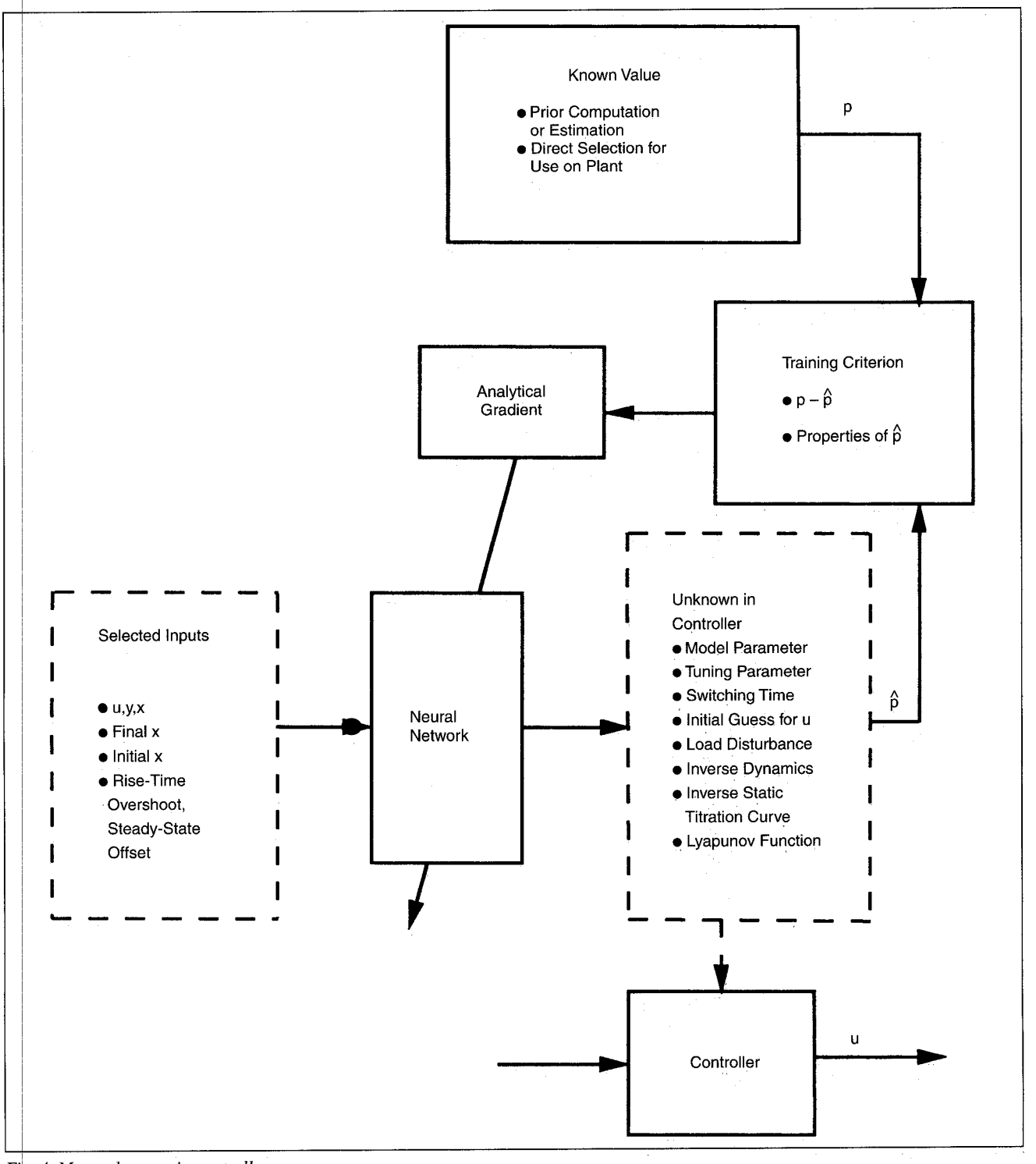


Fig. 4. Map unknowns in controller.

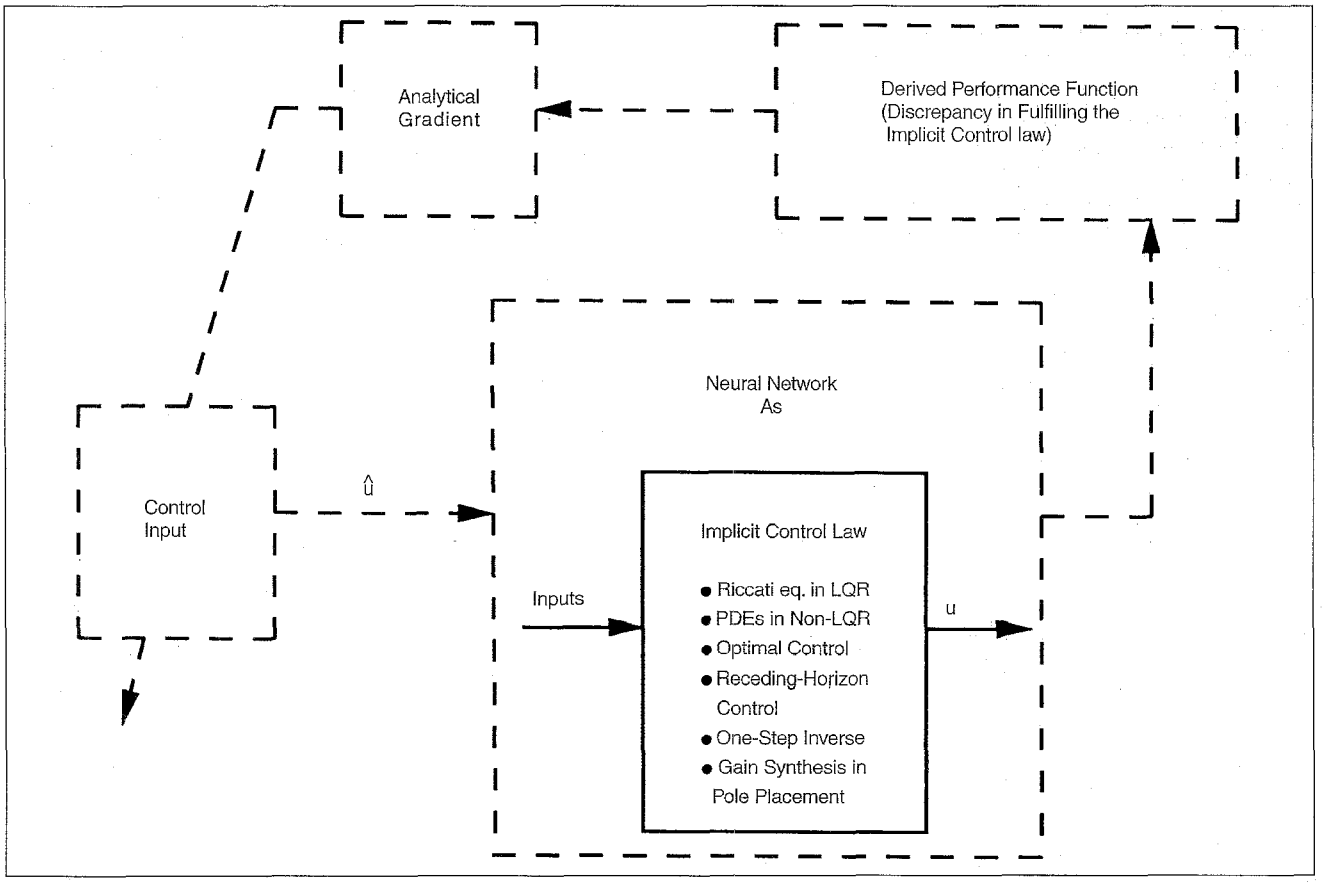


Fig. 5. Solve implicit control law.

model predictions have been used in the trial-and-error optimization of the performance index for, e.g., receding-horizon predictive control of y [31, 38, 39] and x [40], constrained optimal control [41], and trajectory optimization [42]. Optimization of one-step performance indexes through trial-and-error [43-46] or, e.g., through a PI controller at each step [47, 4] is sensitive to unmodeled changing plant dynamics and can lead to closed-loop instability [38], unless a sufficiently conservative gradual [48, 4, 36] or delayed [38] set-point change is used. Inversion of the network model has also been used to compensate for coupling effects in decoupling control [44].

Exact mathematical representation of the network mapping based on its internal structure has been exploited to obtain a stability-based controller [49]. Partial linearization of the network model yields a sensitivity-based, truncated analytical model for use in model-based control schemes [50-53]. The network model also serves as a plant simulation for on-line adaptation of analytical-model parameters in conventional control [54] and for the tuning of, e.g., PID controllers [55].

Aid Modeling Based on Control Goal

In this strategy, neural networks are used to aid models that are based on the ultimate control goal rather than on minimization of the controlled-variable prediction error.

Controlled-Variable Model. Models are ideally developed based on the control goal, since their quality is determined ultimately by the control performance. In this scheme, a network identifying parameters [56] or functions [57-60] in a parent model or directly predicting the controlled variable is trained based on the ultimate control goal (Fig. 3). Exact analytical gradient evaluation then involves the unknown plant in the sensitivity path. To circumvent this problem, crude approximations of the analytical gradient may suffice in some cases, e.g., in stability-based control [56, 58, 59, 60]; otherwise, numerical gradient approximation, e.g., as a finite-difference stochastic approximation or through simultaneous perturbations requiring fewer closed-loop explorations [57], or gradient-free training is used.

Performance-Index Model. In this scheme, a network model mapping input u directly to the performance index J of the control goal is used to determine the optimal control action by iterative optimization. Very general performance indexes are possible, such as minimizing a weighted combination of the rise-time and response overshoot [61]. Open-loop data comprising performance-index values obtained on the actual plant, or on an output model, through prescribed variations in u and other relevant variables, are used to train the network. When a time trajectory is needed as network input, e.g., u and x trajectories for optimal-control performance index, a closed-form parameterization or a staircase-type discretization of the time dependence is used. In case of a parameterized controller, the network maps the controller parameters to the performance index, and the iterative optimization yields optimal tuning of these parameters [62].

Similarly, in dynamic-programming formulations, the recurrence function appearing in the functional equation is mapped by a neural network [63, 64]. Since the recurrence function inherently involves an optimizer, the network training requires stochastic approximation methods in this case [65].

In applications where the control goal delivers only a discrete failure/no-failure verdict, while regarding all output values within the no-failure region as equally desirable, e.g., the angular deviation of an inverted pendulum from the vertical, the network is used to represent a suitable short-term performance index and is trained solely so as to fulfill certain mathematical properties of the latter. Predictions of the short-term performance index are subsequently used to tune a controller meeting the original long-term no-failure objective [66]. When the short-term performance index does not provide an explicit training error, gradient-free learning procedures must be used [67].

Aid Control-Law Implementation

The data storage, retrieval, and interpolation capability and the optimizer role of neural networks aid the implementation of conventional control laws in several ways.

Map Unknowns in Controller. As a convenient interpolation device, the network is trained to store known values of tuning parameters, initial guesses, and intermediate parameters and nonlinearities involved in the control law (Fig. 4). Tedious determination of, e.g., the switching time in optimal control (by extensive calculation) [68], the tuning parameters in PID control (by model-based design following model-parameter estimation) [69], the initial guess in receding-horizon optimization (from previous solutions) [70], or model parameters [71] and nonlinearities [72] in model-based control (using an estimator), is necessary only to provide the "known" values for training and is replaced by the network interpolation in subsequent application.

Training data have been generated also by imposing on the plant chosen values of the network-output variable and recording from the ensuing plant response the resultant values of the network-input variables. This option has been used for networks mapping, e.g., the gain of a proportional controller based on deviations from the desired output characteristics such as rise time, overshoot, or steady-state offset [73]; the load disturbance based on relevant online-measurable quantities [74]; or a dynamic [42] or steady-state [75] inverse model used as a nonlinear transformation in a control algorithm (e.g., the inverse static titration curve in pH control [76]). Known properties of the network-output variable may suffice as training criteria in some cases; e.g., in feedback stabilization of nonlinear affine systems, a network representing the control Lyapunov function that appears in the feedback law is trained solely on the basis of the mathematical properties that define the Lyapunov function [77].

In this scheme, too, using a single network to map multiple variables may entail unnecessary complication of the interpolation surface [78].

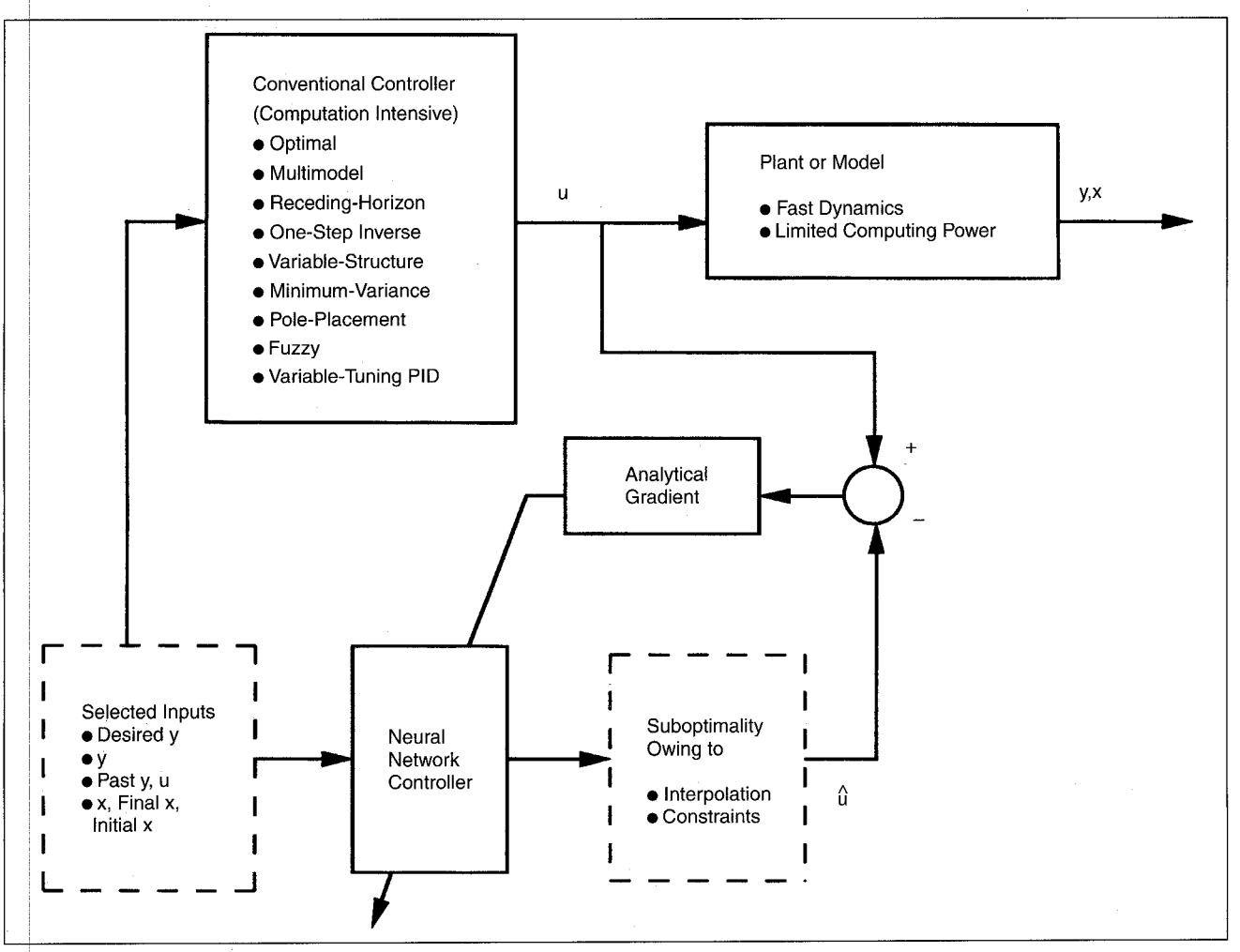


Fig. 6. Mimic a controller.

Solve Implicit Control Law. The role of neural networks as optimizers has been exploited to efficiently solve implicit control laws that otherwise require intensive computational effort. The given control law itself is regarded as a neural network whose input u (instead of the weights) is adjusted by analytical-gradient-based optimization of a derived performance function that is formulated such that its optimum coincides with the solution of the control law (Fig. 5). Applications have appeared, e.g., to solve an algebraic Riccati equation for an infinite-horizon linear quadratic regulator [18], to solve partial differential equations representing the control law for an infinite-horizon nonlinear quadratic regulator [79], to solve finite-horizon non-

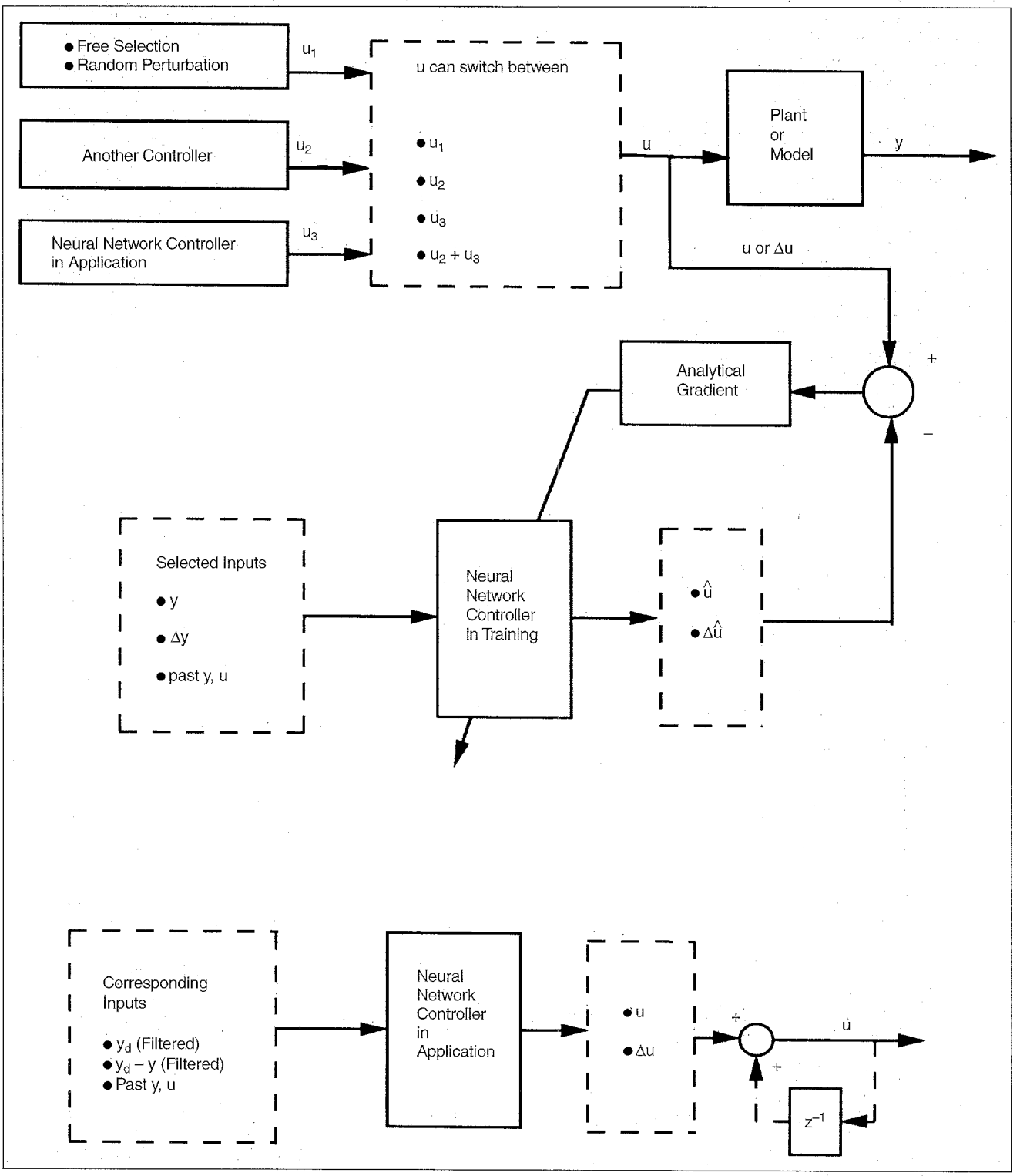


Fig. 7. Use open-loop data.

linear optimal-control problems with inequality constraints [80], to solve an optimal-control problem with a very general objective function using a performance-index model [61], to solve a minimum-norm gain synthesis problem in linear pole-placement control [81], and to solve a one-step inversion optimization as a special case of receding-horizon control [82].

Aid Supervisory Action

The pattern-classification ability of neural networks has been exploited to monitor the performance of a lower-level multi-unit control system and to recommend supervisory actions relating to commissioning of, and communication between, the various units. Networks have been used to recommend intermittent model adaptation based on a diagnosis of the closed-loop performance of a model-based controller and to ensure that sufficient signal excitation is available in the recent data before the adaptation is switched on [83]. Parameter values delivered by estimators [84], sensor readings resulting from fault situations [85], inputs and outputs of controllers [86, 87], or simply unit-step responses [88] have been classified using neural networks into categories that dictate switching between different controllers.

Train Based on u

This category comprises schemes where a neural controller is trained using available input signals u. The training data including the u values are obtained either on the real plant, or on a model simulation. The former option yields a controller without requiring explicit modeling effort.

Mimic Human Expert

In this approach, valuable know-how of a human expert in transforming the available information through experience and intuition into decisions of appropriate control actions for poorly defined problems or difficult-to-analyze plants is extracted as a neural-network functional mapping. The resultant neural controller relieves the expert in subsequent application and, by "disregarding" the mistake outliers during its feature extraction, may even yield more consistent performance than an error-prone expert. Coarsely discretized perception of the available information and decision of the control action by the expert are replaced by finer continuous interpolations in the network, so that it may outperform the expert in making fine adjustments, e.g., to maintain a ship on the current heading [89, 90] or to balance an inverted pendulum [91]. Exploratory investigation of this approach on easy-to-solve control problems, e.g., level control in hydraulic systems with various tank profiles [92, 93], does not extrapolate well to applications for complex control problems such as aircraft landing [94], since the network cannot reliably replicate the expert know-how when it is deprived of important information that is used unconsciously by the expert and is not measured explicitly, e.g., the color, odor, and texture of a reaction mixture, or acoustic and kinesthetic perceptions of a fighter pilot.

In a variation of this approach, the human expert must first translate her tacit, intuitive "feel" for the problem-solving into explicit logical rules for task decomposition and control strategy [95]. These rules are then represented as a neural network through direct geometric methods [96, 95] or through pattern-based training [97]. The approach has been applied to such complex control problems as trailer-truck docking and cart-pendulum systems and offers advantages over conventional rule-based control schemes [96]. The network interpolates reasonable decisions also in the intermediate "gray" situations, akin to the weighted averaging of rules in fuzzy-control algorithms [98].

Mimic a Controller

When a given controller requires large computational or tuning effort for each process condition, several closed-loop trajectories generated by it for different process conditions are used to train a neural controller in an open-loop fashion (Fig. 6). In subsequent application, the neural controller provides approximate control actions by interpolation, reducing the computational expense and eliminating the tuning effort. The approach is specially useful for plants having fast dynamics, e.g., robots [99] and pH neutralization [100], or limited on-line computing capacity, e.g., aircraft [101].

Applications have been reported for, e.g., optimal control [91], multimodel optimal control [102], receding-horizon control [100], one-step inverse control [82, 101, 103], time-optimal control [99], variable-structure control [104], adaptive minimum-variance control [105], pole-placement control [52], fuzzy control [106], and variable-tuning PID control [107]. The network inherits only indirectly and sub-optimally any constraints in the parent controller, e.g., bounds on the control action [100] or the settling time [70] in receding-horizon control. *A posteriori* enforcement of bounds on the control action through squashing functions [99] does not restore optimality under constraint.

Use Open-Loop Data

In this approach, the network uses open-loop input-output data, u and y, to identify an inverse mapping of the plant (Fig. 7). The signal y used as network input during training is replaced by the desired y-value y_d during application. For static plants, feedback control is introduced by using differences Δu and Δy between successive patterns during training and replacing Δy by $y_d - y$ as the network input during application to obtain change in u as the network output [33, 97]. Inadequacy of this static mapping for dynamic plants [61] has been countered by preprocessing the network input y_d or $y_d - y$ through a low-order filter [82, 39], by including as additional network inputs past y and u signals [43, 47, 82] and other signals chosen by prior knowledge or trial-and-error [108, 109, 21], or by using feedback networks with internal dynamics [110, 111].

Direct choice of the u-values in the training data, e.g., as a piecewise-constant signal [112] or a ramp signal between actuator limits [113], may have to span a conservatively large u-range when only the expected y-range is known, resulting in an inferior controller for the actual narrower y-range [114]. Generating the u-values with another controller, e.g., a pre-optimized u-trajectory [115], confines the training data to the expected y-range and ensures reasonable plant operation during data generation. This option differs from the scheme of mimicking a controller (see the so-called section) due to use of the measured y instead of the desired (or optimal) y_d in the training data. Generating the u-values with the neural controller itself, where the y-signal used as network input is different for the control and training steps [108], is useful when changing plant dynamics dictate continuous on-line adaptation of the controller [109, 97]. Possible con-

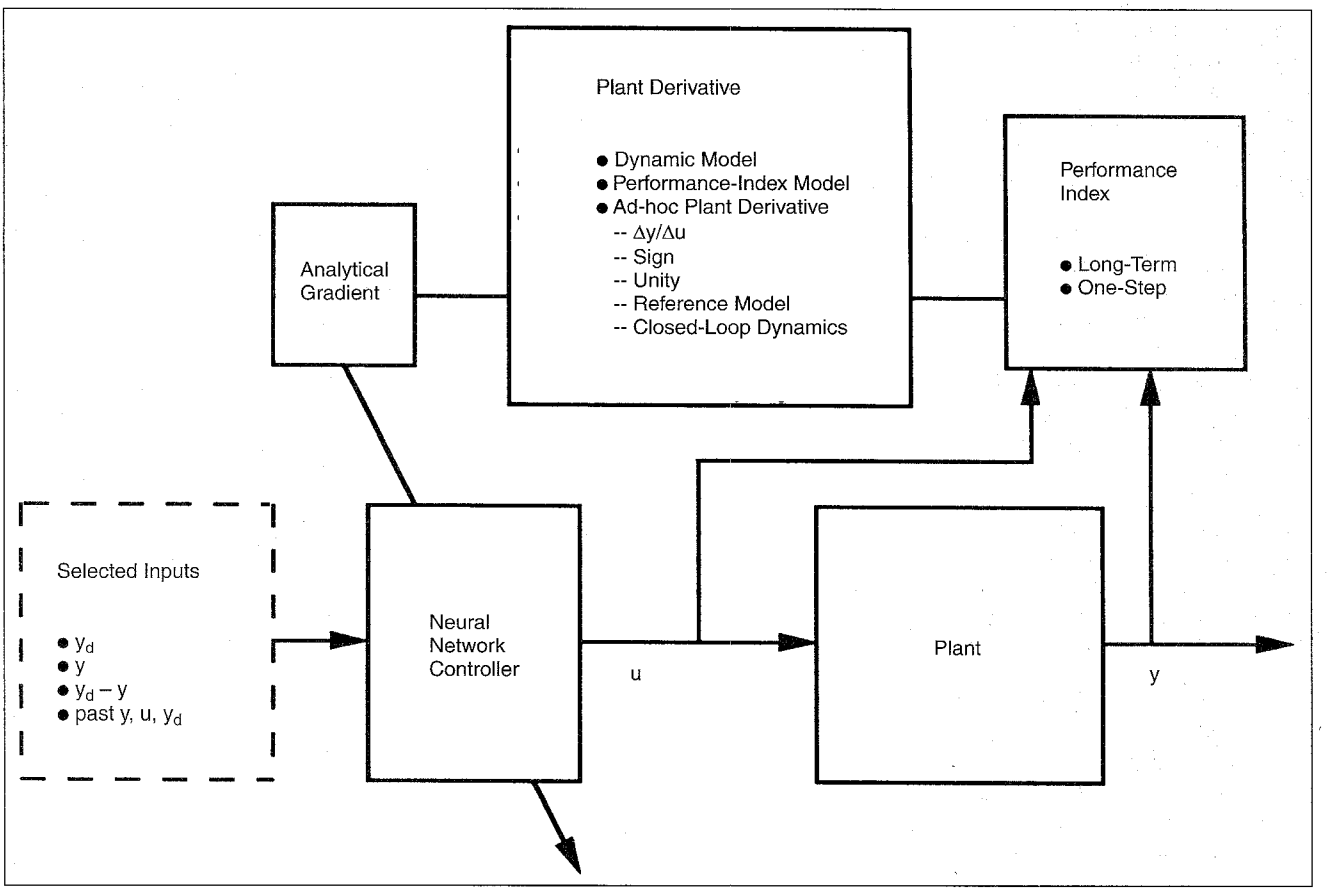


Fig. 8. Train on plant using plant derivative.

vergence of this controller to a single, input-independent network output [114] has been prevented by using a projection procedure for the network weights [43], by generating the u-values as a sum of outputs of the neural controller and another controller in parallel [116, 34], or by pretraining the neural controller using independent u-values [117] (generated, e.g., by random perturbations around an operating point [109] or by another controller [97]) followed by intermittent retraining using the independent u-values [97] or only intermittent updating of the controller weights [109].

In case the plant inverse is not unique, this scheme may approximate a possibly physically inaccessible mean of the non-unique u-values [118]. In case of multiple u-variables, using a separate network for each variable may be advantageous in order to avoid coupling effects during interpolation.

Train Based on Goal

In this category, the neural controller is trained to directly optimize the ultimate control objective and develops the appropriate control strategy without external guidance from exemplary control actions. Conducting the trial-and-error training on a model simulation only requires availability of a sufficiently accurate model, but training on the actual plant entails additional resource consumption, wear-and-tear, and possibly unusable product with each trial.

Train on Model Simulation

Training the neural controller on a model simulation, with subsequent application on the real plant, has the advantage that a possibly large number of needed training trials incurs only computational load, but no plant costs. However, the accuracy of the used model limits the quality of the obtained controller, since the latter is sub-optimal on the plant due to mismatch in the model [30]. The controller training is based invariably on an analytical gradient, which is obtained readily, since the sensitivity path involves the known model instead of the unknown plant. The used simulation model is often itself a neural network [119, 120, 121, 20], which simplifies the sensitivity backpropagation for obtaining the analytical gradient [122]; for unwieldy models, it might be better to use numerical-gradient-based or gradient-free optimization for training.

One-step inverse control is obtained with objectives penalizing the control error at the next sampling instant [118, 123, 103] and possibly the control effort u, the change in u, or violation of bounds on u [124]. A derived one-step objective has been used to train a neural controller to complement (as in the section "Use of Non-Neural Models and Controllers") a linear model-based optimal controller [125]. For control objectives spanning a future time period, e.g., to minimize offset [120] or a more general cost function [119, 126, 121, 20] over a fixed horizon or to reach a target state at the end of a moving horizon [30, 35] or in minimum time [122, 127], the gradient determination requires sensitivity backpropagation through an artificial chain of multiple controller-model pairs arranged in series in the temporal dimen-

sion over that time period. Severe attenuation of the backpropagated signal through a long chain has been countered by using large initial weights and direct connections from network inputs to network outputs [120].

When the current control action should account for the entire future moving target trajectory, a different number of network inputs, and therefore a different network, is needed at each time instant [126]. However, a single network suffices for all time instants when only an end target is included in the objective function [120, 122].

Improved response to setpoint changes has been obtained by using feedforward neural controllers trained as a steady-state inverse of the dynamic simulation model, to generate the zero-offset control action corresponding to any setpoint [121]. For stability-based control objectives, it may be possible to train the controller off-line, based on an *a-priori* obtained training-data set [128].

Train on Plant

Training the neural controller by trial-and-error directly on the plant removes the need for a simulation model and the limitation imposed by the model quality, but entails numerous, possibly expensive, trials on the plant and does not allow straightforward determination of the analytical gradient due to presence of the unknown plant in the sensitivity path. The first three following subsections describe ways to obtain analytical approximations of the gradient, enabling convergence to only a sub-optimal controller.

Plant Derivative from Model. In this scheme, a dynamic model of the plant provides an approximation of the missing plant derivative that is needed for an analytical gradient determination [129, 130] (Fig. 8). Trial-and-error based on this gradient approaches the optimum so long as the inner product of the provided and actual derivatives remains positive [66]. Compared to training the controller on a model simulation (see the section "Train on Model Simulation"), this scheme is more tolerant to model inaccuracies as it utilizes the actual plant output. For dynamic controllers involving feedback connections, the gradient evaluation requires proper backpropagation over unfolded connections [131] or use of a linear sensitivity model that is identified from the dynamic model at each sampling instant [132]. The backpropagation is more straightforward with neural-network models [133-135] than with conventional models [136, 137]; the former are often adapted on-line, simultaneously with the neural-controller training. Performance-index models (see the so-called section) have also been used, providing the derivative of the control objective [66].

The scheme has been applied mainly for inverse control with one-step control objective [113, 138, 21]; less tangible control goals, such as decoupling multivariable plants [32], require a suitably defined objective function. Convergence has been expedited by pretraining the neural controller with open-loop plant data (see the section "Use Open-Loop Data") [113] or by using another controller in parallel (see the section "Use of Non-Neural Models and Controllers") [132].

Ad-Hoc Plant Derivative. Even a crude ad-hoc approximation of the plant derivative may, in many cases, allow convergence sufficiently close to the optimum [139] (Fig. 8). Approximation based on first-order differences uses discrete changes Δy and Δu [140], which are often simply taken from previous iterations [114]. Approximation as only the sign of the plant derivative [76, 132] is often used when the sign is known

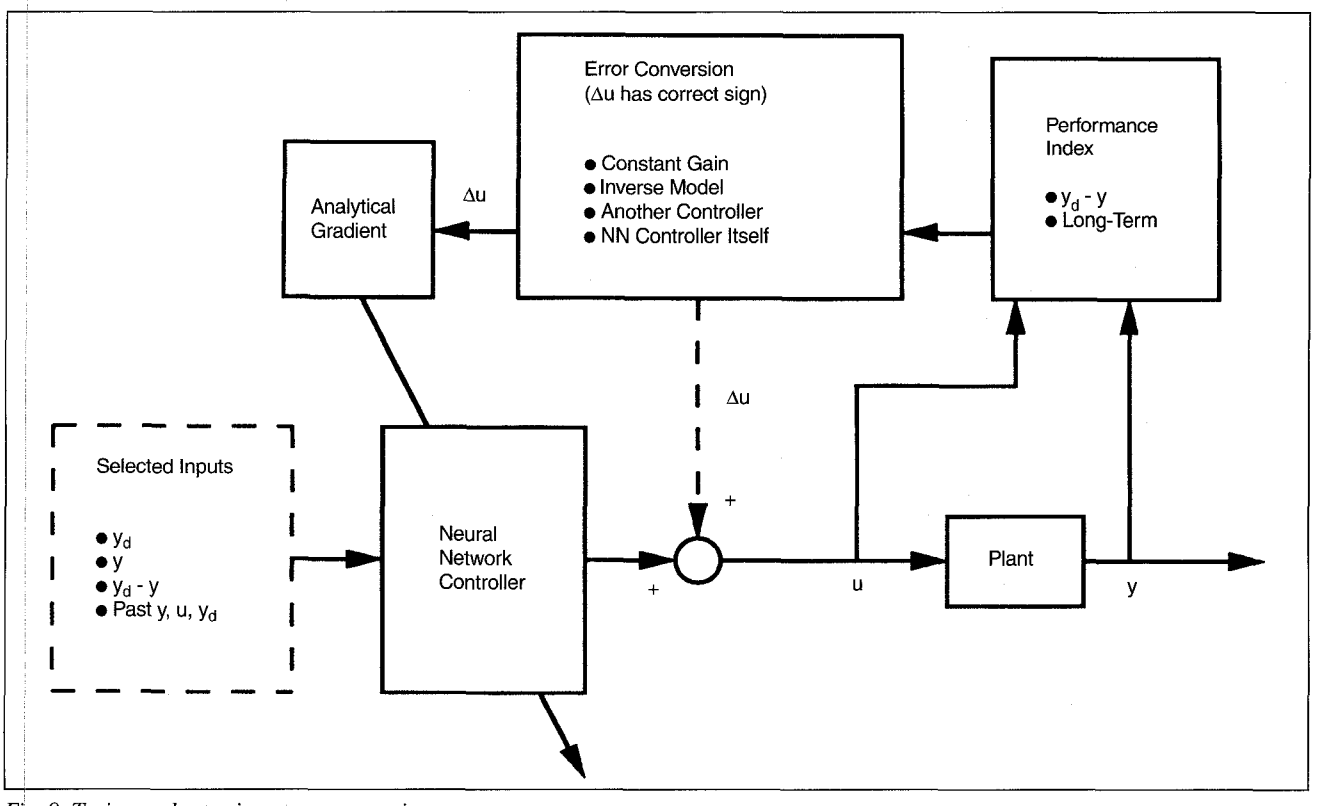


Fig. 9. Train on plant using error conversion.

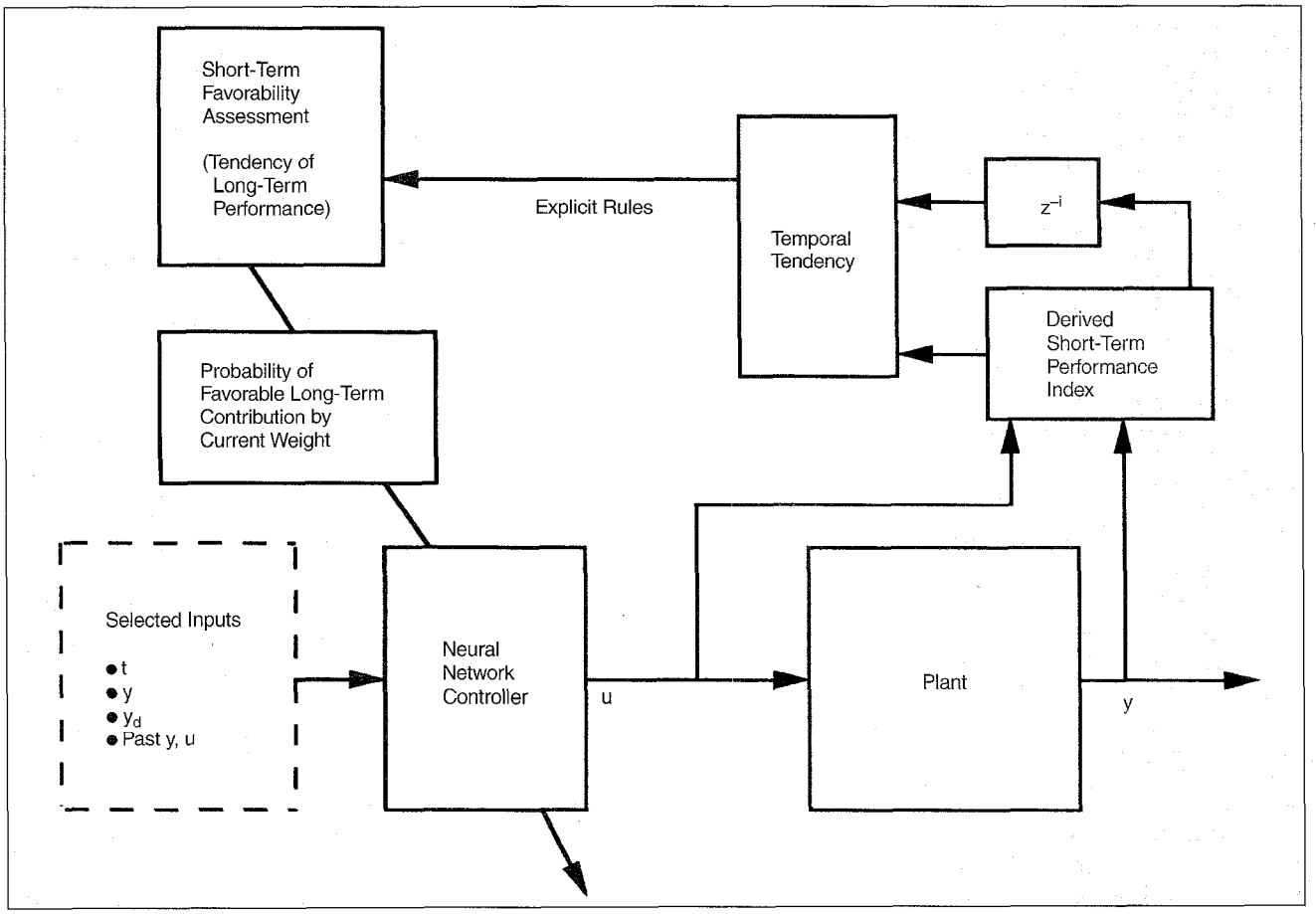


Fig. 10. Gradient-free training on plant using rule-based tendencies.

[141] or is constant [142, 143] and can be estimated on-line. Approximation as unity [129] is possible in the case of a constant and known sign, by embedding the sign directly in the objective function [130]. However, caution is advised against prematurely using the y-error [144, 106] or a scaled y-error [145] as a u-error during training, without formally invoking the unity approximation and ascertaining the prerequisite of a constant and known sign. Another approximation uses the derivative of the reference model that is mimicked by the controller-plant combination in model-reference adaptive control, obtaining a rough estimate of the plant derivative after factoring out the known derivative of the neural controller [133]. Similarly, the plant derivative is extracted from approximate knowledge of the closed-loop dynamics [146]; the latter also suffices to train the neural controller for stability-based objectives [147].

Applications of this strategy have appeared mainly for inverse control with one-step control objective [114, 141, 140, 76]. Local minima possibly caused by the derivative approximation have been escaped by temporarily retraining the neural controller with open-loop plant data (see the section "Use Open-Loop Data") [114].

Error Conversion. In this scheme, the objective function (usually the y-error) is converted into a corresponding approximate u-error Δu that is used directly to train the neural controller (Fig. 9). An approximate conversion to a Δu possessing the correct sign allows approaching the performance-index optimum sufficiently closely in many cases; this property distinguishes the present scheme from those in the section "Train Based on u." The required conversion has been achieved by using a constant gain with appropriate sign [148]; an approximate inverse plant model [149]; a fixed-gain P [6, 150], PD [5], PI [151], or PID [152] controller; a dynamic matrix controller [153]; or the neural controller itself after pretraining with open-loop plant data (see the section "Use Open-Loop Data") [154]. One-step control objectives are most commonly used; general objectives spanning extended time periods require an appropriate controller to provide the necessary conversion of the objective function [153].

The converted Δu provided by another, reliable controller is additionally useful for improving the plant operation during training through correction of the u-signal received by the plant [152]. The conversion and the correction have also been implemented through two separate controllers [155, 148]. When correction is employed, the neural controller begins with negligible output (while the other controller provides the entire control action) and gradually phases out the other controller as it learns to optimize the performance index. This ensures reasonable plant operation during the entire training. Note that here the neural controller does not mimic the other controller (see the section "Mimic a Controller"); it learns a different u-signal than the one output by the other controller.

Numerical Gradient. Numerical approximation of the objective-function gradient with respect to the controller weights requires a large number of trials on the plant when the dimension of the weight vector is large. Stochastic approximation using si-

multaneous perturbations [57] may require fewer plant trials than the standard finite-difference approximation [156]. Objective functions spanning long time periods force a delay in the weight correction [157], requiring more plant trials for convergence than needed for one-step objectives that allow immediate weight correction at each step [57].

Gradient-Free Training. Gradient-free optimization of the objective function with respect to the controller weights using standard search techniques, e.g., optimization over one network weight at a time, stochastic search accepting or rejecting random perturbations in the weight vector [158, 132], stochastic optimization with simultaneous weight increments guided by individual correlations with the objective-function increment [159], stochastic optimization using self-adaptive perturbations in the controller *output* [67], or genetic optimization involving crossovers and mutations of weights [158, 160], may require a large number of trials on the plant, especially when control objectives spanning long time periods delay the performance evaluation and the weight update. Genetic methods, for example, have converged within scores of plant trials only for simple low-order plants and with control objectives spanning up to a score time-steps, e.g., minimum settling time without overshoot [161, 162].

Long-term control objectives, e.g., in optimal control or horizon control, in general, do not allow derivation of an equivalent short-term objective for immediate weight correction without waiting to evaluate the long-term objective; a good current control action for the short-term objective may entail bad consequences later for the long-term objective. This problem has been countered by relying on short-term indicators of the *tendency* of the long-term performance until the latter can be evaluated. For example, in optimal control, probabilities of favorable long-term contribution are assigned to the weights and are updated based on a derived, approximate, short-term performance index spanning a short horizon in the past [163]. The temporal tendency of the short-term performance is translated using explicit rules into a discrete favorable-unfavorable decision or a continuous decision between most favorable and least favorable (Fig. 10). Such explicit rules are available only when the long-term objective involves extremization of a concrete performance measure, e.g., maximization of yield, minimization of energy consumption, or maximization of overall profitability in process applications [110].

Explicit evaluation of the short-term performance as being favorable or unfavorable is difficult when the long-term control objective allows only a success-failure evaluation at the end of a trial, e.g., in constrained operation of a cart-pendulum system where exceeding certain pendulum angles implies nonrecoverable fall and a failed run [164], so that no particular condition is valued more than the other during operation within certain inequality constraints. The evaluation is then performed by a second neural network, called the evaluator, that predicts the eventual performance and allows a short-term favorability as-

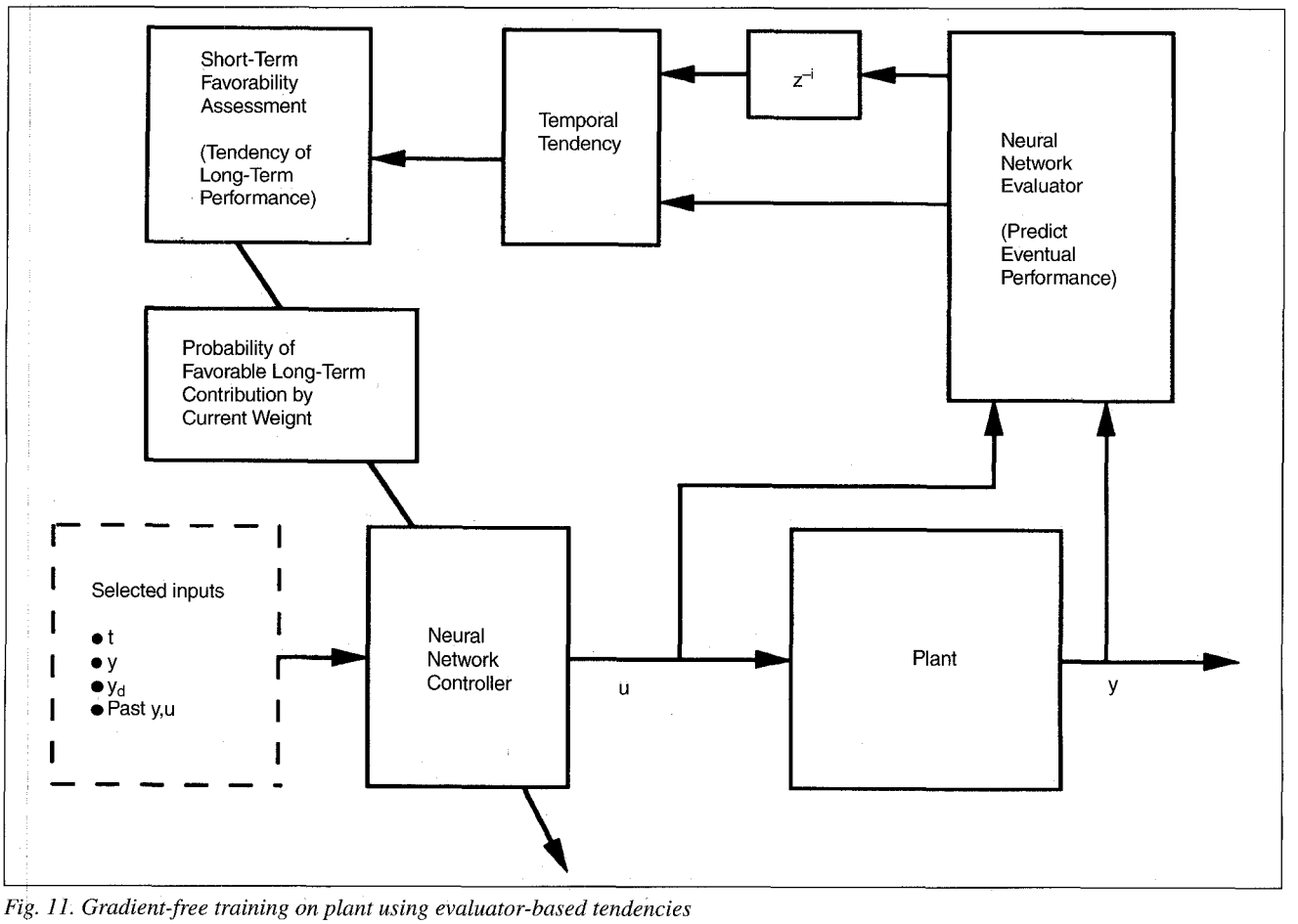


Fig. 11. Gradient-free training on plant using evaluator-based tendencies

sessment through the temporal tendency of these predictions (Fig. 11). The evaluator is trained in a similar manner as, and concurrently with, the controller [67]. This approach is useful in process control, e.g., when the sole control objective is to maintain the final product quality within an acceptable range, and has been used also with extremizing control objectives by transforming them into inequality constraints through insight into the plant behavior, e.g., a simple regulator objective is reformulated as appropriately qualified "desirability" bounds on the controlled variable [165]. Combining the evaluator and controller functions into a single network, with neurons performing a dual role, has led to faster convergence and better performance for the cart-pendulum system [166].

General Enhancement Features

Various features for improving the above control schemes have been used, each of which is applicable to several branches of the classification tree in Fig. 1. The most common of such features are summarized below.

Disturbance Rejection and Time Delays

In any scheme that employs a neural or non-neural forward process model, errors due to plant-model mismatch and unmodeled disturbances are alleviated, as in conventional control, by estimating the model error through comparison of past predicted and measured y-values. This estimate provides a correction for the future predictions delivered by the model [100, 40] and for the current error signal input to the controller [44, 54], either directly [4] or through a filter designed to enhance noise rejection and robustness [46]. Additionally, the setpoint or the error signal is preprocessed through a filter designed for desired closed-loop dynamics [44, 112, 103]. These two separate filters for model-error rejection and setpoint dynamics are tuned individually [82, 167] or combined into a single filter to form the well-known internal model control structure [47, 39].

Known time delays entail feeding appropriately delayed inputs into the network models and controllers [108, 123, 167], leading to the Smith-predictor compensation together with the correction based on the filtered model error [82, 55]. Unknown time delays are detected by trial-and-error based on the estimated model error or the control performance [97]. The above features have been employed mainly in the schemes of sections "Predict Controlled Variable," "Use Open-Loop Data," and "Train on Model Simulation," but are applicable to other schemes also.

On-Line Adaptation and Multiple Networks

On-line retraining of the neural-network model or controller, or use of multiple networks [168, 169], or both [170], may be useful in adapting to changing plant characteristics. On-line training involves issues such as specification of an adaptation rate that is compatible with the time-variation of the plant [52, 159], selection of a performance threshold at which the adaptation switches on and off [61, 48], addition of a carefully selected excitation signal [39, 52], use of devices such as parameter projection [167], and slower updating of the controller than of the model [123, 171], whereby stability considerations play an important role.

On-line adaptation of the controller weights becomes necessary also whenever the setpoint changes and is not available to the controller as a network input [123, 130, 137]. Controller adaptation is usually inadvisable for the schemes of sections "Ad-Hoc Plant Derivative" and "Error Conversion," since the training strategy in these schemes makes sense only asymptotically [159]. When the control objective involves a moving horizon, controller adaptation based on a past horizon may be useful [171].

Feedback Network Structure

Dynamic models and controllers are implemented more efficiently using network structures that involve various kinds of signal feedback. Feedback connections between all the nodes [36, 135] involve unnecessary complexity that may be reduced by using feedback connections only within each layer of internal nodes [155, 138, 151], only from the network outputs [27], only to the network inputs [138, 53], or only locally for each neuron [154, 131, 55, 49]. Feedback of the network output to the network input is also commonly used to obtain dynamic models [36, 37] and controllers [47, 134, 76].

The lag of the feedback signals must be chosen carefully in order for them to be useful [155]. Feedback connections further complicate issues relating to stability and learning rate during on-line adaptation [55, 159]. Certain feedback-network models are amenable to direct exact linearization for controller design [27].

Use of Non-Neural Models and Controllers

A conventional model or controller may be utilized in several ways to relieve the burden on the neural-network model or controller [172]. The neural-network part may cover the unmodeled effects in the conventional model or controller [125, 153, 147, 54]; conversely, a parallel conventional controller may reduce or eliminate the offset due to an inadequate neural controller [39, 101]. A neural controller may, in all schemes, be parameterized through the structure of a conventional controller, so that the network outputs certain parameters such as gains [129, 163, 154] or nonlinear functions [139, 21] that build the overall neural controller together with the conventional control structure imposed on it; the scheme in Fig. 3 represents a special case of this general possibility. The design of, and the mapping represented by, the neural-network model or controller is rendered more tangible to human insight and interpretation by deliberately imposing if-then rule-based fuzzy reasoning on the structure of the network [173, 98].

Conclusion

The presented classification reveals the fundamental features distinguishing various network-based control schemes. It shows that schemes regarded in the literature as being similar are often actually intrinsically different. For example, subsets of eight inherently distinct schemes—those in sections "Predict Controlled Variable," "Solve Implicit Control Law," "Mimic a Controller," "Use Open-Loop Data," "Train on Model Simulation," "Plant Derivative from Model," "Ad-Hoc Plant Derivative," and "Error Conversion"—are commonly lumped together as one-step inverse control. Similarly, closely related schemes are sometimes regarded as being basically different. For example, various subsets of the section "Use Open-Loop Data" tend to get placed in different categories. Often a proposed scheme includes several secondary features, especially those in the section "General Enhancement Features," which makes it less transparent for an as-

sessment of its novelty. For example, one scheme [82] proposes a neural controller as in the section "Mimic a Controller" that mimics the inversion of a neural model from the section "Predict Controlled Variable" using a neural optimizer as in the section "Solve Implicit Control Law," and another scheme [97] combines a subset of the section "Mimic Human Expert" with several features from the sections "Use Open-Loop Data" and "General Enhancement Features." The presented classification is not necessarily complete and is certainly not closed; certain past work possibly missed by the author as well as future control schemes may well necessitate new branches at any level of the classification tree.

References

[1] B. Widrow, D.E. Rumelhart, and M.A. Lehr, "Neural Networks: Applications in Industry, Business and Science," *Journal A*, vol. 35, no. 2, pp. 17-27, 1994.

[2] J.D. Boskovic and K.S. Narendra, "Comparison of Linear, Nonlinear and Neural-Network-Based Adaptive Controllers for a Class of Fed-Batch Fermentation Processes," *Automatica*, vol. 31, no. 6, pp. 817-840, 1995.

[3] P. Dutta and R.R. Rhinehart, "Experimental Comparison of a Novel, Simple, Neural Network Controller and Linear Model Based Controllers," in *Proc. American Control Conf., (Seattle, WA), pp. 1787-1789, 1995.*

[4] M.J. Piovoso, K.A. Kosanovich, V. Rokhlenko, and A. Guez, "A Comparison of Three Nonlinear Controller Designs Applied to a Non-Adiabatic First-Order Exothermic Reaction in a CSTR," in *Proc. American Control Conf.*, (Chicago, IL), pp. 490-494, 1992.

[5] R.E. Nordgren and P.H. Meckl, "An Analytical Comparison of a Neural Network and a Model-Based Adaptive Controller," *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 685-694, 1993.

[6] L.G. Kraft and D.P. Campagna, "A Comparison Between CMAC Neural Network Control and Two Traditional Adaptive Control Systems," *IEEE Control Systems Magazine*, vol. 10, pp. 36-43, 1990.

[7] P.J. Werbos, "An Overview of Neural Networks for Control," *IEEE Control Systems Magazine*, vol. 11, no. 1, pp. 40-41, 1991.

[8] K.J. Hunt, D. Sbarbaro, R. Zbikowski, and P.J. Gawthrop, "Neural Networks For Control Systems—A Survey," *Automatica*, vol. 28, pp. 1083-1112, 1992.

[9] A.G. Barto, "Connectionist Learning For Control," in *Neural Networks for Control* (W.T. Miller, R.S. Sutton, and P.J. Werbos, eds.), pp. 9-61, Cambridge, MA: MIT Press, 1990.

[10] J. Thibault and B.P.A. Grandjean, "Process Control Using Feedforward Neural Networks," *J. Syst. Eng.*, vol. 2, pp. 198-212, 1992.

[11] P.J.G. Lisboa, "Neural Networks For Control: Evolution, Revolution or Renaissance," in *Application of Neural Networks to Modelling and Control* (G.F.Page, J.B.Gomm, and D.Williams, eds.), pp. 13-24, London, UK: Chapman & Hall, 1993.

[12] M. Brown and C.J. Harris, "Neural Networks for Modelling and Control," in *Advances in Intelligent Control* (C.J.Harris, ed.), pp. 17-55, London: Taylor & Francis, 1994.

[13] J.A.K. Suykens and H. Bersini, "Neural Control Theory: An Overview," *Journal A*, vol. 37, no. 3, pp. 4-10, 1996.

[14] E.F. Camacho and M.R. Arahal, "Adaptive Control and Neural Networks," *Journal A*, vol. 37, no. 2, pp. 16-21, 1996.

[15] H. Tolle, P.C. Parks, E. Ersu, M. Hormel, and J. Militzer, "Learning Control with Interpolating Memories—General Ideas, Design Lay-Out, Theoretical Approaches and Practical Applications," *Int. J. Control*, vol. 56, pp. 291-317, 1992.

[16] R.C. Eberhart, "Standardization of Neural Network Terminology," *IEEE Trans. Neural Networks*, vol. 1, no. 2, pp. 244-245, 1990.

[17] C. Zhu and F.W. Paul, "Self-Tuning Adaptive Control Using Fourier Series Neural Networks," in *Proc. American Control Conf.*, (San Francisco, CA), pp. 2524-2528, 1993.

[18] K.L. Moore, D.S. Naidu, and M. Siddaiah, "A Real-Time Adaptive Linear Quadratic Regulator Using Neural Networks," in *Proc. European Control Conf.*, (Groningen, NL), pp. 805-809, 1993.

[19] J.A.K. Suykens, B.L.R. De Moor, and J. Vandewalle, "Nonlinear System Identification Using Neural State Space Models, Applicable to Robust Control Design," *Int. J. Control*, vol. 62, no. 1, pp. 129-152, 1995.

[20] J. Mazak, A.A.H. Damen, and A.C.P.M. Backx, "Neural State Transition Controller for a Polymerization Reactor," *Journal A*, vol. 37, no. 3, pp. 34-39, 1996.

[21] S. Bittanti and L. Piroddi, "GMV Techniques for Nonlinear Control with Neural Networks," *IEE Proc.-Control Theory Appl.*, vol. 141, no. 2, pp. 57-69, 1994.

[22] M.M. Polycarpou, "Stable Adaptive Neural Control Scheme for Nonlinear Systems," *IEEE Trans. Automat.Contr.*, vol. 41, no. 3, pp. 447-451, 1996.

[23] F.-C. Chen and H.K. Khalil, "Adaptive Control of a Class of Nonlinear Discrete-Time Systems Using Neural Networks," *IEEE Trans. Automat.Contr.*, vol. 40, no. 5, pp. 791-801, 1995.

[24] A. Karakasoglu, S.I. Sudharsanan, and M.K. Sundareshan, "Identification and Decentralized Adaptive Control Using Dynamical Neural Networks with Application to Robotic Manipulators," *IEEE Trans. Neural Networks*, vol. 4, no. 6, pp. 919-930, 1993.

[25] A. Yesildirek and F.L. Lewis, "Feedback Linearization Using Neural Networks," *Automatica*, vol. 31, no. 11, pp. 1659-1664, 1995.

[26] D.J. Cooper, L. Megan, and R.F. Hinde, "Comparing Two Neural Networks for Pattern Based Adaptive Process Control," *AIChE J*, vol. 38, no. 1, pp. 41-55, 1992.

[27] M. Nikolaou and V. Hanagandi, "Control of Nonlinear Dynamical Systems Modeled by Recurrent Neural Networks," *AIChE J*, vol. 39, no. 11, pp. 1890-1894, 1993.

[28] A. Delgado, C. Kambhampati, and K. Warwick, "Dynamic Recurrent Neural Network for System Identification and Control," *IEE Proc.-Control Theory Appl.*, vol. 142, no. 4, pp. 307-314, 1995.

[29] W.-S. Lin, H.-Y. Shue, and C.-H. Wang, "A Neural Network Approach of Input-Output Linearization of Affine Nonlinear Systems," in *Proc. American Control Conf.*, (Baltimore, MD), pp. 2933-2937, 1994.

[30] A.U. Levin and K.S. Narendra, "Control of Nonlinear Dynamical Systems Using Neural Networks: Controllability and Stabilization," *IEEE Trans. Neural Networks*, vol. 4, no. 2, pp. 192-206, 1993.

[31] J. Saint-Donat, N. Bhat, and T.J. McAvoy, "Neural Net Based Model Predictive Control," *Int. J. Control*, vol. 54, no. 6, pp. 1453-1468, 1991.

[32] K.S. Narendra and S. Mukhopadhyay, "Adaptive Control of Nonlinear Multivariable Systems Using Neural Networks," *Neural Networks*, vol. 7, no. 5, pp. 737-752, 1994.

[33] H.-H. Huang and H.-P. Wang, "Tandem Neural Networks for Welding Process Control," *J. Syst. Eng.*, vol. 2, pp. 295-303, 1992.

[34] L. Xu, J.-P. Jiang, and J. Zhu, "Supervised Learning Control of a Nonlinear Polymerisation Reactor Using the CMAC Neural Network for Knowledge Storage," *IEE Proc.-Control Theory Appl.*, vol. 141, pp. 33-38, 1994.

[35] A.U. Levin and K.S. Narendra, "Control of Nonlinear Dynamical Systems Using Neural Networks—Part II: Observability, Identification, and Control," *IEEE Trans. Neural Networks*, vol. 7, no. 1, pp. 30-42, 1996.

[36] L. Jin, P.N. Nikiforuk, and M.M. Gupta, "Adaptive Control of Discrete-Time Nonlinear Systems Using Recurrent Neural Networks," *IEE Proc.-Control Theory Appl.*, vol. 141, no. 3, pp. 169-176, 1994.

[37] B. Schenker and M. Agarwal, "Predictive Control of a Bench-Scale Chemical Reactor Based on Neural-Network Models," *IEEE Trans. Control Syst. Tech.* (accepted).

[38] E. Hernandez and Y. Arkun, "Study of The Control-Relevant Properties of Backpropagation Neural Network Models of Nonlinear Dynamical Systems," *Computers chem. Engng*, vol. 16, no. 4, pp. 227-240, 1992.

[39] D. Sbarbaro, "Control of a pH Plant Using Connectionist Representations," in *Proc. 12th IFAC World Congress*, vol. 10, (Sydney, Australia), pp. 377-380, 1993.

[40] M.J. Willis, G.A. Montague, C. di Massimo, M.T. Tham, and A.J. Morris, "Artificial Neural Networks in Process Estimation and Control," *Automatica*, vol. 28, no. 6, pp. 1181-1187, 1992.

[41] Q. Chen and W.A. Weigand, "Dynamic Optimization of Nonlinear Processes By Combining Neural Net Model with UDMC," *AIChE J*, vol. 40, no. 9, pp. 1488-1497, 1994.

[42] Y. Wada and M. Kawato, "A Neural Network Model for Arm Trajectory Formation Using Forward and Inverse Dynamics Models," *Neural Networks*, vol. 6, pp. 919-932, 1993.

[43] B.E. Ydstie, "Forecasting and Control Using Adaptive Connectionist Networks," *Computers chem. Engng*, vol. 14, no. 4/5, pp. 583-599, 1990.

[44] G.M. Scott and W.H. Ray, "Experiences with Model-Based Controllers Based on Neural Network Process Models," *J. Process Control*, vol. 3, no. 3, pp. 179-195, 1993.

[45] L. Behera, M. Gopal, and S. Chaudhury, "Inversion of RBF Networks and Applications to Adaptive Control of Nonlinear Systems," *IEE Proc.-Control Theory Appl.*, vol. 142, no. 6, pp. 617-624, 1995.

[46] A.C. van Ommeren and B. Roffel, "Neural Network Based Control of Chemical Processes," *Journal A*, vol. 37, no. 3, pp. 46-51, 1996.

[47] K.J. Hunt and D. Sbarbaro, "Neural Networks for Nonlinear Internal Model Control," *IEE Proc.-Control Theory Appl.*, vol. 138, no. 5, pp. 431-438, 1991.

[48] D.A. Hoskins, J.N. Hwang, and J. Vagners, "Iterative Inversion of Neural Networks and Its Application to Adaptive Control," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 292-301, 1992.

[49] G.A. Rovithakis and M.A. Christodoulou, "Adaptive Control of Unknown Plants Using Dynamical Neural Networks," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 24, no. 3, pp. 400-412, 1994.

[50] X. Ma and R.N.K. Loh, "Neural Network-Based Successive One-Step-Ahead Control of Nonlinear Systems," in *Proc. American Control Conf.*, (Baltimore, MD), pp. 2129-2133, 1994.

[51] G. Lightbody and G.W. Irwin, "A Novel Neural Internal Model Control Structure," in *Proc. American Control Conf.*, (Seattle, WA), pp. 350-354, 1995.

[52] M.S. Ahmed and I.A. Tasadduq, "Neural-Net Controller for Nonlinear Plants: Design Approach Through Linearisation," *IEE Proc.-Control Theory Appl.*, vol. 141, no. 5, pp. 315-322, 1994.

[53] O. Sorensen, "Non-Linear Pole-Placement Control with a Neural Network," *European J. Control*, vol. 2, no. 1, pp. 36-43, 1996.

[54] A. Draeger, S. Engell, and H. Ranke, "Model Predictive Control Using Neural Networks," *IEEE Control Systems Magazine*, vol. 15, no. 5, pp. 61-66, 1995.

[55] Y. Tan and R. De Keyser, "Auto-Tuning PID Control Using Neural Predictor to Compensate Large Time-Delay," in *Proc. IEEE Conf. Control Applications*, (Glasgow, UK), pp. 1429-1434, 1994.

[56] J.D. Leon, E.N. Sanchez, and A. Chataigner, "Mechanical System Tracking Using Neural Networks," in *Proc. American Control Conf.*, (Baltimore, MD), pp. 2555-2559, 1994.

[57] J.C. Spall and J.A. Cristion, "Direct Adaptive Control of Nonlinear Systems Using Neural Networks and Stochastic Approximation," in *Proc. 31st IEEE Conf. Decision Control, (Tucson, AZ), pp. 878-883, 1992.*

[58] S.-W. Lee and J.-H. Kim, "CMAC Network-Based Robust Controller For Systems with Friction," in *Proc. 34th IEEE Conf. Decision Control, (New Orleans, LA), pp. 2938-2939, 1995.*

[59] A.J.N. van Breemen and L.P.J. Veelenturf, "Neural Adaptive Feedback Linearization Control," *Journal A*, vol. 37, no. 3, pp. 65-71, 1996.

[60] S. Jagannathan and F.L. Lewis, "Multilayer Discrete-Time Neural-Net Controller with Guaranteed Performance," *IEEE Trans. Neural Networks*, vol. 7, no. 1, pp. 107-130, 1996.

[61] Y.-H. Pao, S.M. Phillips, and D.J. Sobajic, "Neural-Net Computing and the Intelligent Control of Systems," *Int. J. Control*, vol. 56, no. 2, pp. 263-289, 1992.

[62] J.J.L. Lin, D.S.H. Wong, and S.W. Yu, "Optimal Multiloop Feedback Design Using Simulated Annealing and Neural Network," *AIChE J*, vol. 41, no. 2, pp. 430-434, 1995.

[63] S. Schaal and C.G. Atkeson, "Robot Juggling: Implementation of Memory-Based Learning," *IEEE Control Systems Magazine*, vol. 14, pp. 57-71, 1994.

[64] D.V. Prokhorov, R.A. Santiago, and D.C. Wunsch, "Adaptive Critic Designs: A Case Study For Neurocontrol," *Neural Networks*, vol. 8, no. 9, pp. 1367-1372, 1995.

[65] R.S. Sutton, A.G. Barto, and R.J. Williams, "Reinforcement Learning Is Direct Adaptive Optimal Control," *IEEE Control Systems Magazine*, vol. 12, pp. 19-22, 1992.

[66] V. Gullapalli, "Direct Associative Reinforcement Learning Methods for Dynamic Systems Control," *Neurocomputing*, vol. 9, pp. 271-292, 1995.

[67] V. Gullapalli, J.A. Franklin, and H. Benbrahim, "Acquiring Robot Skills Via Reinforcement Learning," *IEEE Control Systems Magazine*, vol. 14, pp. 13-24, 1994.

[68] R.W. Anderson and V. Vemuri, "Neural Networks Can Be Used for Open-Loop, Dynamic Control," *Int. J. Neural Networks*, vol. 3, no. 3, pp. 71-83, 1992.

[69] M. Roele and K. Warwick, "The Use of Artificial Intelligence in Self-Tuning PID Controllers," *J. Syst. Eng.*, vol. 5, pp. 223-238, 1995.

[70] D. Sbarbaro and K.J. Hunt, "A Nonlinear Receding Horizon Controller Based on Connectionist Models," in *Proc. 30th IEEE Conf. Decision Control*, (Brighton, England), pp. 172-173, 1991.

[71] G. Geng and G.M. Geary, "A Neural Network Based RLS Parameter Estimation Algorithm and Its Application in Predictive Control," in *Proc. IEEE Int. Symp. Intelligent Control*, (Chicago, IL), pp. 127-131, 1993.

[72] D.J. Linse and R.F. Stengel, "A System Identification Model for Adaptive Nonlinear Control," in *Proc. American Control Conf.*, (Boston, MA), pp. 1752-1757, 1991.

[73] A.J. Lawrence and C.J. Harris, "A Label-Driven CMAC Intelligent Control Strategy," in *Application of Neural Networks to Modelling and Control*

(G.F.Page, J.B.Gomm, and D.Williams, eds.), pp. 89-103, London, UK: Chapman & Hall, 1993.

[74] C. Askew and M.K. Sundareshan, "Adaptive Variable Structure Control of Flexible Manipulators by Neural Network Payload Identification," in *Proc. 32nd IEEE Conf. Dec. Contr.*, (San Antonio, TX), pp. 3249-3250, 1993.

[75] S. Ramchandran and R.R. Rhinehart, "A Very Simple Structure for Neural Network Control of Distillation," *J. Process Control*, vol. 5, no. 2, pp. 115-128, 1995.

[76] A.P. Loh, K.O. Looi, and K.F. Fong, "Neural Network Modelling and Control Strategies for a pH Process," *J. Process Control*, vol. 5, no. 6, pp. 355-362, 1995.

[77] Y. Long and M.M. Bayoumi, "Feedback Stabilization: Control Lyapunov Functions Modelled by Neural Networks," in *Proc. 32nd IEEE Conf. Decision Control, (San Antonio, TX), pp. 2812-2814, 1993.*

[78] M.A. Sartori and P.J. Antsaklis, "A Gaussian Neural Network Implementation for Control Scheduling," in *Proc. IEEE Int. Symp. Intelligent Control*, (Arlington, VA), pp. 400-404, 1991.

[79] C.J. Goh, "On The Nonlinear Optimal Regulator Problem," *Automatica*, vol. 29, no. 3, pp. 751-756, 1993.

[80] R.-L. Yang and C.-P. Wu, "Neural Networks for Exact Solution of Constrained Optimal Control Problems," in *Proc. American Control Conf.*, (Baltimore, MD), pp. 1379-1383, 1994.

[81] J. Wang and G. Wu, "A Multilayer Recurrent Neural Network For On-Line Synthesis of Minimum-Norm Linear Feedback Control Systems Via Pole Assignment," *Automatica*, vol. 32, no. 3, pp. 435-442, 1996.

[82] E.P. Nahas, M.A. Henson, and D.E. Seborg, "Nonlinear Internal Model Control Strategy for Neural Network Models," *Computers chem. Engng*, vol. 16, no. 12, pp. 1039-1057, 1992.

[83] R.F. Hinde and D.J. Cooper, "Using Pattern Recognition in Controller Adaptation and Performance Evaluation," in *Proc. American Control Conf.*, (San Francisco, CA), pp. 74-78, 1993.

[84] S. Leonhardt, J. Bushardt, R. Rajamani, K.Hedrick, and R.Isermann, "Parameter Estimation of Shock Absorbers with Artificial Neural Networks," in *Proc. American Control Conf.*, (San Francisco, CA), pp. 716-720, 1993.

[85] H.E. Rauch, "Intelligent Fault Diagnosis and Control Reconfiguration," *IEEE Control Systems Magazine*, vol. 14, pp. 6-12, 1994.

[86] L. Megan and D.J. Cooper, "A Neural Network Strategy for Disturbance Pattern Classification and Adaptive Multivariable Control," *Computers chem. Engng*, vol. 19, no. 2, pp. 171-186, 1995.

[87] K. KrishnaKumar and R. Lattus, "Control Surface Failure Detection and Accommodation Using Neuro-Controllers," *Neural Comput & Applic*, vol. 2, pp. 120-128, 1994.

[88] A. Leva and L. Piroddi, "Model-Specific Autotuning of Classical Regulators: a Neural Approach to Structural Identification," *Control Eng. Practice*, vol. 4, no. 10, pp. 1381-1391, 1996.

[89] G.A. Pugh, "Ship Directional Control by Synthetic Neural Network," in *Proc. American Control Conf.*, (San Diego, CA), pp. 3028-3029, 1990.

[90] Y.M. Enab, "Intelligent Controller Design for the Ship Steering Problem," *IEE Proc.-Control Theory Appl.*, vol. 143, no. 1, pp. 17-24, 1996.

[91] T. Frohlinghaus, A.Weichert, and P.Rujan, "Hierarchical Neural Networks for Time-Series Analysis and Control," *Network: Computation in Neural Systems*, vol. 5, pp. 101-116, 1994.

[92] Y.-H. Pao and D.J. Sobajic, "Nonlinear Process Control with Neural Nets," *Neurocomputing*, vol. 2, pp. 51-59, 1990.

[93] A. Prochazka and M. Storek, "Nonlinear System Control Using Neural Networks," *Neural Network World*, vol. 3, pp. 261-266, 1993.

[94] C. Jorgenson and C. Schley, "A Neural Network Baseline Problem for Control of Aircraft Flare and Touchdown," in *Neural Networks for Control* (W.T.Miller, R.S.Sutton, and P.J.Werbos, eds.), pp. 403-425, Cambridge, MA: MIT Press, 1990.

[95] R.E. Jenkins and B.P. Yuhas, "A Simplified Neural Network Solution Through Problem Decomposition: The Case of the Truck Backer-Upper," *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 718-719, 1993.

[96] J. Hao and J. Vandewalle, "A Rule-Based Neural Controller for Inverted Pendulum System," *Int. J. Neural Systems*, vol. 4, no. 1, pp. 55-64, 1993.

[97] M. Ishida and J. Zhan, "A Policy- and Experience-Driven Neural Network and Its Application to Nonlinear Process Control," in *Proc. European Control Conf.*, (Groningen, NL), pp. 471-474, 1993.

[98] J. Nie and D.A. Linkens, "FCMAC: A Fuzzified Cerebellar Model Articulation Controller With Self-Organizing Capacity," *Automatica*, vol. 30, no. 4, pp. 655-664, 1994.

[99] C.J. Goh, N.J. Edwards, and A.Y. Zomaya, "Feedback Control of Minimum-Time Optimal Control Problems Using Neural Networks," *Optim. Control Appl. Methods*, vol. 14, pp. 1-16, 1993.

[100] M. Pottmann and D.E. Seborg, "A Nonlinear Predictive Control Strategy Based on Radial Basis Function Networks," in *Proc. IFAC Symp. DYCORD+'92*, (College Park, MD), pp. 309-314, 1992.

[101] J.E. Steck, K. Rokhsaz, and S.-P. Shue, "Linear and Neural Network Feedback for Flight Control Decoupling," *IEEE Control Systems Magazine*, vol. 16, no. 4, pp. 22-30, 1996.

[102] M.A. Al-Akhras, G.M. Aly, and R.J. Green, "Neural Network Learning Approach of Intelligent Multimodel Controller," *IEE Proc.-Control Theory Appl.*, vol. 143, no. 4, pp. 395-400, 1996.

[103] F.-S. Ho and P. Ioannou, "Traffic Flow Modeling and Control Using Artificial Neural Networks," *IEEE Control Systems Magazine*, vol. 16, no. 5, pp. 16-26, 1996.

[104] J.X. Xu, J. Donne, and U. Ozguner, "Synthesis of Feedback Linearization and Variable Structure Control with Neural Net Compensation," in *Proc. IEEE Int. Symp. Intelligent Control*, (Arlington, VA), pp. 184-189, 1991.

[105] G.-J. Wang and D.K. Miu, "Unsupervised Adaptive Neural-Network Control of Complex Mechanical Systems," in *Proc. American Control Conf.*, (Boston, MA), pp. 28-29, 1991.

[106] H.M. Mashaly, A.M. Sharaf, M. Mansour, and A.A. El-Sattar, "A Photovoltaic Maximum Power Tracking Using Neural Networks," in *Proc. IEEE Conf. Control Applications*, (Glasgow, UK), pp. 167-172, 1994.

[107] R. Burns and R. Richter, "A Neural-Network Approach to the Control of Surface Ships," *Control Eng. Practice*, vol. 4, no. 3, pp. 411-416, 1996.

[108] K. Khorasani and R.V. Patel, "A Neural Network Architecture for Model Reference Adaptive Control," in *Proc. 29th IEEE Conf. Decision Control, (Honolulu, HI), pp. 2768-2769, 1990.*

[109] D.C. Kennedy and V.H. Quintana, "Neural Network Regulators for Synchronous Machines," in *Proc. 12th IFAC World Congress*, vol. 5, (Sydney, Australia), pp. 131-136, 1993.

[110] L.H. Ungar, B.A. Powell, and S.N. Kamens, "Adaptive Networks for Fault Diagnosis and Process Control," *Computers chem. Engng*, vol. 14, no. 4/5, pp. 561-572, 1990.

[111] J.E. Cramer and B.F. Womack, "Adaptive Control Using Neural Networks," in *Proc. American Control Conf.*, (Boston, MA), pp. 681-686, 1991.

[112] Y. Hashimoto, T. Katoh, T. Shiina, A. Yoneya, Y. Togari, and C. McGreavy, "Piecewise Linear Controller Improving Its Own Reliability," *J. Process Control*, vol. 6, no. 2/3, pp. 129-136, 1996.

[113] M. Khalid, S. Omatu, and R. Yusof, "MIMO Furnace Control with Neural Networks," *IEEE Trans. Control Syst. Tech.*, vol. 1, no. 4, pp. 238-245, 1993.

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

[115] A. Kuan and B. Bavarian, "Compensation of Unstructured Uncertainty in Manipulators Using Neural Networks," in *Proc. 31st IEEE Conf. Dec. Contr.*, (Tucson, AZ), pp. 2706-2709, 1992.

[116] J.G. Kuschewski, S. Hui, and S.H. Zak, "Application of Feedforward Neural Networks to Dynamical System Identification and Control," *IEEE Trans. Control Syst. Tech.*, vol. 1, no. 1, pp. 37-49, 1993.

[117] H.C. Andersen, F.C. Teng, and A.C. Tsoi, "Single Net Indirect Learning Architecture," *IEEE Trans. Neural Networks*, vol. 5, no. 6, pp. 1003-1005, 1994.

[118] D. Lowe, "On The Iterative Inversion of RBF Networks: A Statistical Interpretation," in *Proc. IEE 2nd Int. Conf. Artificial Neural Networks*, (Bournemouth, UK), pp. 29-33, 1991.

[119] D.H. Nguyen and B. Widrow, "Neural Networks for Self-Learning Control Systems," *Int. J. Control*, vol. 54, no. 6, pp. 1439-1451, 1991.

[120] D.C. Cressy, I.T. Nabney, and A.M. Simper, "Neural Control of a Batch Distillation," *Neural Comput & Applic*, vol. 1, pp. 115-123, 1993.

[121] Y.-M. Park, M.-S. Choi, and K.Y. Lee, "An Optimal Tracking Neuro-Controller for Nonlinear Dynamic Systems," *IEEE Trans. Neural Networks*, vol. 7, no. 5, pp. 1099-1110, 1996.

[122] T.R. Niesler and J.J. du Plessis, "Time-Optimal Control by Means of Neural Networks," *IEEE Control Systems Magazine*, vol. 15, pp. 23-33, 1995.

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

[124] R.M. Sanner and D.L. Akin, "Neuromorphic Pitch Attitude Regulation of an Underwater Telerobot," *IEEE Control Systems Magazine*, vol. 10, pp. 62-67, 1990.

[125] Y. Iiguni, H. Sakai, and H. Tokumaru, "A Nonlinear Regulator Design in the Presence of System Uncertainties Using Multilayered Neural Networks," *IEEE Trans. Neural Networks*, vol. 2, no. 4, pp. 410-417, 1991.

[126] T. Parisini and R. Zoppoli, "Neural Networks for Feedback Feedforward Nonlinear Control Systems," *IEEE Trans. Neural Networks*, vol. 5, no. 3, pp. 436-449, 1994.

[127] E.S. Plumer, "Optimal Control of Terminal Processes Using Neural Networks," *IEEE Trans. Neural Networks*, vol. 7, no. 2, pp. 408-418, 1996.

[128] S. Yu and A.M. Annaswamy, "Control of Nonlinear Dynamic Systems Using a Stability Based Neural Network Approach," in *Proc. 34th IEEE Conf. Decision Control, (New Orleans, LA), pp. 1290-1295, 1995.*

[129] S.-L. Chang and S.S. Nair, "A New Neural Network Control Architecture for a Class of Nonlinear Dynamic Systems," in *Proc. American Control Conf.*, (San Francisco, CA), pp. 79-83, 1993.

[130] R.J. Hampo and K.A. Marko, "Investigation of the Application of Neural Networks to Fault Tolerant Control of an Active Suspension System," in *Proc. American Control Conf.*, (Chicago, IL), pp. 11-13, 1992.

[131] P.S. Sastry, G. Santharam, and K.P. Unnikrishnan, "Memory Neuron Networks for Identification and Control of Dynamical Systems," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 306-319, 1994.

[132] G. Lightbody and G.W. Irwin, "Direct Neural Model Reference Adaptive Control," *IEE Proc.-Control Theory Appl.*, vol. 142, no. 1, pp. 31-43, 1995.

[133] M.S. Ahmed, "Block Partial Derivative and Its Application to Neural-Net-Based Direct-Model-Reference Adaptive Control," *IEE Proc.-Control Theory Appl.*, vol. 141, no. 5, pp. 305-314, 1994.

[134] C.-C. Ku and K.Y. Lee, "Diagonal Recurrent Neural Networks for Dynamic Systems Control," *IEEE Trans. Neural Networks*, vol. 6, no. 1, pp. 144-156, 1995.

[135] T. Chovan, T. Catfolis, and K. Meert, "Neural Network Architecture for Process Control Based on the RTRL Algorithm," *AIChE J*, vol. 42, no. 2, pp. 493-502, 1996.

[136] T. Ishikawa, J. Tsuji, H. Ohmori, and A. Sano, "Novel Configuration of Nonlinear Adaptive Control Incorporating Neural Network," in *Proc. 12th IFAC World Congress*, vol. 9, (Sydney, Australia), pp. 483-488, 1993.

[137] F. Beaufays, Y. Abdel-Magid, and B. Widrow, "Application of Neural Networks to Load-Frequency Control in Power Systems," *Neural Networks*, vol. 7, no. 1, pp. 183-194, 1994.

[138] G.V. Puskorius and L.A. Feldkamp, "Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter Trained Recurrent Networks," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 279-297, 1994.

[139] N. Sadegh, "A Perceptron Network for Functional Identification and Control of Nonlinear Systems," *IEEE Trans. Neural Networks*, vol. 4, no. 6, pp. 982-988, 1993.

[140] K.P. Venugopal and S.M. Smith, "Improving The Dynamic Response of Neural Network Controllers Using Velocity Reference Feedback," *IEEE Trans. Neural Networks*, vol. 4, no. 2, pp. 355-357, 1993.

[141] M. Saerens and A. Soquet, "Neural Controller Based on Back-Propagation Algorithm," *IEE Proc.-F*, vol. 138, no. 1, pp. 55-62, 1991.

[142] W.H. Schiffmann and H.W. Geffers, "Adaptive Control of Dynamic Systems by Back Propagation Networks," *Neural Networks*, vol. 6, pp. 517-524, 1993.

[143] Y. Zhang, P. Sen, and G.E. Hearn, "An On-Line Trained Adaptive Neural Controller," *IEEE Control Systems Magazine*, vol. 15, pp. 67-75, 1995.

[144] D.H. Rao and M.M. Gupta, "Dynamic Neural Adaptive Control Schemes for Linear and Nonlinear Systems," in *Proc. American Control Conf.*, (San Francisco, CA), pp. 1450-1454, 1993.

[145] F.L. Lewis, A. Yesildirek, and K. Liu, "Multilayer Neural-Net Robot Controller with Guaranteed Tracking Performance," *IEEE Trans. Neural Networks*, vol. 7, no. 2, pp. 388-399, 1996.

[146] S. Jung and T.C. Hsia, "A New Neural Network Control Technique for Robot Manipulators," in *Proc. American Control Conf.*, (Seattle, WA), pp. 878-882, 1995.

[147] A.J. Calise, B.S. Kim, J. Leitner, and J.V.R. Prasad, "Helicopter Adaptive Flight Control Using Neural Networks," in *Proc. 33rd IEEE Conf. Decision Control, (Lake Buena Vista, FL), pp. 3336-3341, 1994.*

[148] D. Dong, W.N. White, and H. Luo, "Investigation of Kinematics and Inverse Dynamics Algorithm with a DSP Implementation of a Neural Network," in *Proc. American Control Conf.*, (Baltimore, MD), pp. 2460-2464, 1994.

[149] A.M.S. Zalzala and A.S. Morris, "A Neural Network Approach to Adaptive Robot Control," *Int. J. Neural Networks*, vol. 2, no. 1, pp. 17-35, 1991.

[150] S.J. Hepworth and A.L. Dexter, "Neural Control of Non-Linear HVAC Plant," in *Proc. IEEE Conf. Control Applications*, (Glasgow, UK), pp. 1849-1854, 1994.

[151] T. Shibata and T. Fukuda, "Hierarchical Intelligent Control for Robotic Motion," *IEEE Trans. Neural Networks*, vol. 5, no. 5, pp. 823-832, 1994.

[152] H. Gomi and M. Kawato, "Neural Network Control for a Closed-Loop System Using Feedback-Error-Learning," *Neural Networks*, vol. 6, pp. 933-946, 1993.

[153] M. Lee and S. Park, "A New Scheme Combining Neural Feedforward Control with Model-Predictive Control," *AIChE J*, vol. 38, no. 2, pp. 193-200, 1992.

[154] A. Karakasoglu and M.K. Sundareshan, "A Recurrent Neural Network-Based Adaptive Variable Structure Model-Following Control of Robotic Manipulators," *Automatica*, vol. 31, no. 10, pp. 1495-1507, 1995.

[155] R.T. Newton and Y. Xu, "Neural Network Control of a Space Manipulator," *IEEE Control Systems Magazine*, vol. 13, pp. 14-22, 1993.

[156] M. Bahrami and K.E. Tait, "Model Reference Direct Adaptive Control of Nonlinear Plants Using Neural Networks," *Int. J. Neural Systems*, vol. 5, no. 1, pp. 77-82, 1994.

[157] C. James Li, L. Yan, and N.W. Chbat, "Powell's Method Applied to Learning Neural Control of Three Unknown Dynamic Systems," *Optim. Control Appl. Methods*, vol. 16, pp. 251-262, 1995.

[158] S. Geva and J. Sitte, "A Cartpole Experiment Benchmark for Trainable Controllers," *IEEE Control Systems Magazine*, vol. 13, pp. 40-51, 1993.

[159] K.P. Venugopal, A.S. Pandya, and R. Sudhakar, "A Recurrent Neural Network Controller and Learning Algorithm for the On-Line Learning Control of Autonomous Underwater Vehicles," *Neural Networks*, vol. 7, no. 5, pp. 833-846, 1994.

[160] S. Jain, P.-Y. Peng, A. Tzes, and F. Khorrami, "Neural Network Designs with Genetic Learning for Control of a Single Link Flexible Manipulator," in *Proc. American Control Conf.*, (Baltimore, MD), pp. 2570-2574, 1994.

[161] Y. Ichikawa and T. Sawa, "Neural Network Application for Direct Feedback Controllers," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 224-231, 1992.

[162] S. Park, L.-J. Park, and C.H. Park, "A Neuro-Genetic Controller for Nonminimum Phase Systems," *IEEE Trans. Neural Networks*, vol. 6, no. 5, pp. 1297-1300, 1995.

[163] Q.H. Wu and A.C. Pugh, "Reinforcement Learning Control of Unknown Dynamic Systems," *IEE Proc.-Control Theory Appl.*, vol. 140, no. 5, pp. 313-322, 1993.

[164] A.G. Barto, R.S. Sutton, and C.W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 13, no. 5, pp. 835-846, 1983.

[165] J.C. Hoskins and D.M. Himmelblau, "Process Control Via Artificial Neural Networks and Reinforcement Learning," *Computers Chem. Engng*, vol. 16, no. 4, pp. 241-251, 1992.

[166] J.S. Morgan, E.C. Patterson, and A.H. Klopf, "Drive-Reinforcement Learning: a Self-Supervised Model for Adaptive Control," *Network: Computation in Neural Systems*, vol. 1, pp. 439-448, 1990.

[167] H. Koivisto, V.T. Ruoppila, and H.N. Koivo, "Real-Time Neural Network Control—An IMC Approach," in *Proc. 12th IFAC World Congress*, vol. 4, (Sydney, Australia), pp. 47-52, 1993.

[168] R.A. Jacobs and M.I. Jordan, "A Modular Connectionist Architecture for Learning Piecewise Control Strategies," in *Proc. American Control Conf.*, (Boston, MA), pp. 1597-1602, 1991.

[169] K.S. Narendra and A.U. Levin, "Regulation of Nonlinear Dynamical Systems Using Multiple Neural Networks," in *Proc. American Control Conf.*, (Boston, MA), pp. 1609-1614, 1991.

[170] W.K. Tan and T.H. Lee, "A Neural Net Control System With Parallel Adaptive Enhancements," in *Proc. American Control Conf.*, (Chicago, IL), pp. 61-62, 1992.

[171] K. KrishnaKumar, S. Rickard, and S. Bartholomew, "Adaptive Neuro-Control for Spacecraft Attitude Control," *Neurocomputing*, vol. 9, no. 2, pp. 131-148, 1995.

[172] M. Agarwal, "Combining Neural and Conventional Paradigms for Modeling, Prediction, and Control," *Int. J. Systems Sci.*, vol. 28, no. 1, pp. 65-81, 1997.

[173] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On Fuzzy Modeling Using Fuzzy Neural Networks with the Back-Propagation Algorithm," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 801-806, 1992.

Mukul Agarwal received the bachelor's degree from the Indian Institute of Technology, Kanpur, in 1978, the master's degree from the University of Delaware, Newark, in 1980, and the doctoral degree from the University of California at Santa Barbara in 1987. Since 1988, he has been with the Systems Engineering Group at the Swiss Federal Institute of Technology, Zurich, where he has been a senior lecturer since 1990. He has been a visiting researcher at the Lund Institute of Technology, Sweden, and has consulted for Procter and Gamble Co., Cincinnati, OH. From 1980 to 1982, he was a project engineer with Autodynamics, Inc., Freehold, where he designed and tested analog simulations of large-scale plants including control networks. His recent research has focused on nonlinear estimation, optimal control, batch processes, and combining neural networks and conventional techniques.