Parameter Estimation for Process Control With Neural Networks

Tariq Samad and Anoop Mathur
Honeywell SSDC, Minneapolis, Minnesota

ABSTRACT

Neural networks are applied to the problem of parameter estimation for process systems. Neural network parameter estimators for a given parametrized model structure can be developed by supervised learning. Training examples can be dynamically generated by using a process simulation, resulting in trained networks that are capable of high generalization. This approach can be used for a variety of parameter estimation applications. A proof-of-concept open-loop delay estimator is described, and extensive simulation results are detailed. Some results of other parameter estimation networks are also given. Extensions to recursive and closed-loop identification and application to higher-order processes are discussed.

KEYWORDS: ***neural networks, process control, parameter estimation, system identification, delay compensation, neurocontrol***

1. INTRODUCTION

Artificial neural networks, as a technology, are both distinctive and diversified. In the domain of control systems, several ways to effectively use neural networks have been identified (Barto [1], Samad [2]). These include direct and inverse process modeling, controller modeling, and controller development through reinforcement learning or model-based optimization.

In general, most applications of neural networks to control problems have focused on modeling nonlinear processes (e.g., Bhat and McAvoy [3], Narendra and Parthasarathy [4]). The need for techniques to adequately deal with nonlinear processes is unquestioned. Real plants are always nonlinear, yet current control technology is rooted in linear systems. This

Address correspondence to Tariq Samad, Honeywell SSDC, 3660 Technology Drive, Minneapolis, MN 55418.

Received August 1, 1991; accepted February 26, 1992.

International Journal of Approximate Reasoning 1992; 7:149–164

655 Avenue of the Americas, New York, NY 10010 0888-613X/92/$5.00

line of research has the potential to contribute significantly to the future practice of control.

However, notable advantages accrue from the assumption of linearity. In many practical applications, any degradation of control that may result from assuming a linear plant is comparatively insignificant. In particular, with linear models, process identification can be "one-shot": Given any initial state, the system response to an input stimulus such as a step or ramp function provides complete information regarding system behavior at any initial state and to any arbitrary input signal. Thus process modeling requires exactly one sequence of observations. In contrast, if the linearity constraint is not imposed on the model form, system responses to a large number of different input stimuli at different operating points will have to be collected and analyzed.

There are substantial near-term benefits to be gained by using neural networks in control applications. Even in linear systems, many problems arise in identification and control that require the use of nonlinear techniques for their solution. In this paper, we discuss the use of neural networks for an important aspect of process system identification: parameter estimation for continuous-time models. Section 2 gives a brief overview of system identification approaches. Section 3 discusses our approach for neural-network-based parameter estimation. Section 4 describes in detail a proof-of-concept application. Sections 5 and 6 outline some extensions, and we conclude with a summary. An earlier version of this paper appeared as Samad and Mathur [5].

2. SYSTEM IDENTIFICATION APPROACHES

System identification refers to the problem of determining a mathematical expression of system behavior based on data from the system. It is thus sometimes distinguished from "modeling," in which mathematical models are developed on the basis of process knowledge and not experimentation (Ljung [6]). System identification is a critical problem in modern process control. Most advanced control algorithms are model-based and require a mathematical model of the process. There are two tasks that must be accomplished before a system can be identified:

1. Process characterization. A functional form of the system being modeled is determined. The form usually has several associated parameters.
2. Parameter estimation. Values for the parameters of the form adopted are computed.

An important distinction for process characterization is between discrete-time and continuous-time models. In both cases, once the model

form is specified (usually by a user or through some heuristic methods), curve-fitting techniques are typically employed to estimate model parameters.

2.1. Discrete-Time Models

In modern control systems, discrete-time models predominate. Here the process output is modeled as a linear function of sampled process history. A popular approach is the autoregressive moving-average (ARMA) model:

$$y(t) = \sum_{i=1}^{k} a_i y(t-i) + \sum_{i=1}^{l} b_i u(t-i) \tag{1}$$

where $y(t)$ and $u(t)$ are the process output and input, respectively, at time t and the a_i and b_i are model parameters. Additional terms can be incorporated to model process disturbances. Once k and l are determined, parameter estimation reduces to an exercise in linear algebra or, alternatively, in optimization in a quadratic surface. In either case, well-known techniques with guaranteed convergence and stability properties exist. A major disadvantage of such models is that the parameters are not physically meaningful; the relationship between the coefficients of Eq. (1) and important characteristics of a process such as its gains, time constants, and delays is indirect. An ARMA model is of little help for process understanding. The use of such "black-box" models (Ljung [6]) also does not allow convenient incorporation of prior knowledge of a process, say for model structure selection.

2.2. Continuous-Time Models

Continuous-time models of linear systems can be expressed either in the frequency domain (via the Laplace transform) or in the time domain (or state space representation). For example, a time domain representation of the simplest dynamical system, a linear first-order process, is

$$\frac{dy(t)}{dt} = -\frac{1}{\tau} y(t) + \frac{K}{\tau} u(t) \tag{2}$$

In the above formulation, the model parameters are physically meaningful. K represents the steady-state gain of the process, and τ is the time constant governing the dynamical behavior. However, the problem of estimating the values of these parameters from process input–output data is considerably more difficult: the parameters are nonlinear functions of process data. Thus, despite their appeal, the identification of continuous-time models is not commonly attempted.

There is an important parameter that we have not considered above: the process delay. In both discrete- and continuous-time models, delay estima-

tion is a nonlinear problem. In discrete-time model identification, ad hoc methods are usually employed.

3. ESTIMATING PHYSICAL PARAMETERS FROM PROCESS DATA

As mentioned above, it is desirable to compute physical parameters for a process. In order to do so from process data, a nonlinear modeling technique is required. We have been investigating the application of feedforward neural networks trained by using backpropagation (Werbos, [7], Rumelhart et al. [8]) for this problem. In our case, a training example consists of sampled process input and output vectors (which together constitute the training input) and the associated process parameters (the desired output). Based on these examples, the network implements a function for estimating parameters from process input/output (Figure 1). The input parameters of the function are successive time samples of

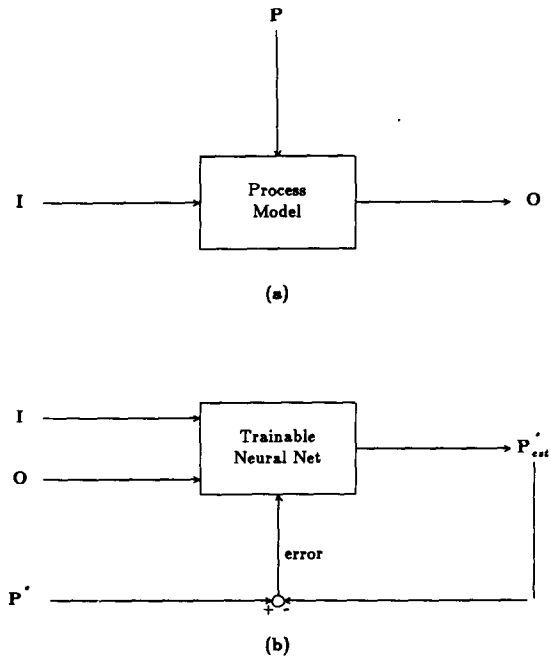

Figure 1. (a) Generation and (b) use of training data. **I**, process input vector; **O**, process output vector; **P**, values for model parameters; **P***, values for model parameters to be estimated (a subset of **P**); $\mathbf{P}^*_{est}$, estimates of **P***. The error signal is used to train the network.

process input and output; the outputs of the function are estimates of desired process parameters.

In most applications of backpropagation, training data have to be obtained before learning is initiated. Static training sets are therefore used —the same training examples are presented repeatedly until the error over them is sufficiently small. There are two potential problems with this approach. First, "overlearning" is often observed—instead of averaging out noise in the training data, the network can learn to fit the data with undesirably high accuracy. Second, the training data may not be sufficiently rich to enable accurate generalization.

In most cases, there is no feasible alternative to the use of a predefined training set. In our approach, however, there is. Recall that we are attempting to estimate values for process *model* parameters. A process model is used for generating training data, and we can therefore generate training data at will (and at relatively small computational expense). We have adopted the extreme case: a new training example is generated for every training iteration. The process parameter values for a training example are randomly chosen from within predetermined ranges. If the prior distributions of parameter values are known, these distributions can be followed for training data generation. Otherwise, some reasonable distribution (e.g., Gaussian or uniform) can be used. The dynamic generation of training samples is computationally more expensive than repeated use of a static training set, but we can expect the expense to be outweighed by the increased accuracy of the learned network and the better sampling of the parameter space that will thereby be achieved. This approach is equivalent to generating, in advance, a sufficiently large training set. However, this is a post factum equivalence. Dynamic generation of examples assumes no knowledge of what "sufficiently large" is for an application.

We also adopt the common practice of deliberately corrupting the training data with some amount of (Gaussian) noise. We have observed that this results in better estimation.

This approach for developing parameter estimators is general-purpose. It can be used for closed-loop or open-loop identification, for estimating any and all model parameters, and for linear or nonlinear process models. The generality of the approach derives from the universal approximation capabilities of neural networks (Hornik et al. [9]). Given an appropriate network architecture, any nonlinear mapping can be realized. However, the utility of this result is limited by several caveats. The network size may be too large to be computationally tractable, the convergence time for the learning algorithm may be intolerably long, and a number of trials may have to be conducted, with different initial conditions, before convergence to a useful solution is achieved. Thus practical considerations can render

possible applications infeasible, and it may often be useful to adopt simplifications to the extent the domain permits. For example, many (although not all) system identification applications are open-loop ones. In this case, the input perturbation can be identical for all training examples and need not be provided as input to the network. The constraint this imposes is that the same input perturbation, or (if the process model is linear) a scaled version of it, must be used in the field (i.e., with the trained network). In practice, system identification is often done on the basis of step inputs to a process. A network trained on step responses would thus have reasonably broad utility.

We have investigated the estimation of several parameters, including the process gain and the process time constant for first-order and second-order processes and the damping constant for second-order processes. These investigations were successful but were not pursued intensively. Most of our effort has instead been directed toward one specific system identification application, delay identification, discussed next. Good estimates for process delay are difficult to obtain and yet are essential for accurate control.

4. PROOF-OF-CONCEPT: A NEURAL NETWORK DELAY IDENTIFIER

To demonstrate the concept of neural-network-based parameter estimation, we have developed a prototype delay identification tool. More precisely, we have developed a neural network delay identifier for the open-loop estimation of process delays for a linear first-order process model (with delay). First-order processes with delay form an important class of models because they can be used to model some complex processes with reasonable accuracy.

Linear first-order processes with delay are modeled by the equation

$$\frac{d}{dt}y(t) = -\frac{1}{\tau_p}y(t) + \frac{K_p}{\tau_p}u(t - \theta)$$

where $y(t)$ is the process response, τ_p is the time constant of the process, K_p is the process gain, and θ is the process delay. K_p, τ_p, and θ are the parameters of the model.

A training example is generated using a simulation of the process model (Figure 2). The model, with its parameters assigned to values within predetermined ranges, is given a step input. The process output is sampled at some predetermined rate, and the resulting real-valued vector is used as the training input. The process delay value that was used to generate the process output is the "desired output."

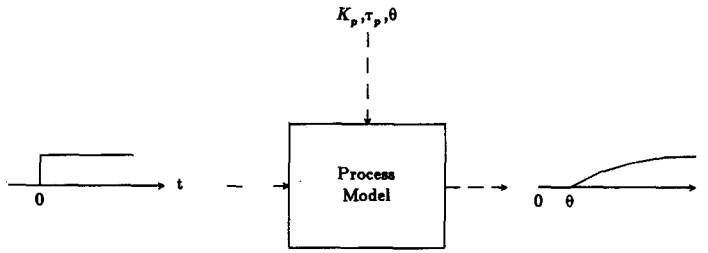

Figure 2. Generation of training data for delay identification.

It should be noted that the identification is in terms of samples, not absolute units of time. By changing the sampling rate, the range of delays that can be identified (by the same trained network) can be controlled. Thus a network trained to identify time delays of between 0 and 500 sec from 50 samples collected at 10-sec intervals could be used, without any further training, to identify time delays of up to 5 hr (say) from 50 samples collected at 6-min intervals. The minimum delay that can be identified is determined by how soon after the step input the process output sampling is started. If the minimum process delay is known to be n sec, we can start sampling n sec after the step input is given. The desired network output would then be $\theta - n$, and we would add n to the output of the trained network to obtain the estimated process delay.

4.1. Simulation Results

We have run numerous simulations of training and operation to evaluate the proof-of-concept system. Here we discuss one in detail.

The ranges of parameters considered were as follows: τ_p, 10–200 sec; θ, 0–500 sec; K_p, 0.5–1.5. Uniform distributions were used for all ranges. Training on a range of K_p values is not strictly necessary if the correct value is available in operation. As the process model is linear, the process output can easily be normalized. However, without training on a range of K_p, even small changes in the value of this parameter can result in significant error in delay estimation. The noise in training inputs was Gaussian with 99% (3 standard deviations) falling within $\pm 5\%$ of the range of process output values, which was normalized between 0 and 1. We later describe the results of a simulation with 50% noise. The output of the process was sampled every 10 sec after the step input was given. Fifty samples were collected and used as input to the network, which had 50 input units (one for each sample), 15 hidden units, and one output unit (the delay estimate). The network was trained on 6 million examples. The

value of the backpropagation learning rate parameter η was 0.1 for the first 1 million training iterations and 0.01 thereafter.

After training, the network was tested on new (also randomly generated) data. Tests were performed to determine the accuracy of delay identification as a function of delay, as a function of τ_p, as a function of K_p, and as a function of the amount of (Gaussian) noise.

Figures 3–5 depict the results of various tests. Each of these graphs shows the estimation error over a range of values of a particular parame-

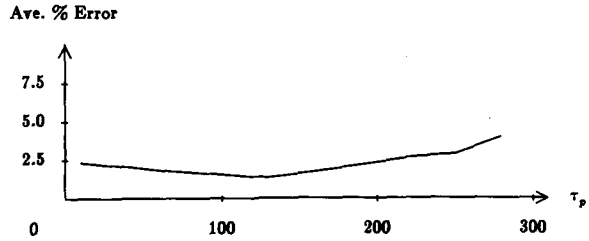

Figure 3. Error in delay identification as a function of τ_p. θ (actual) = 250 sec; K_p = 1.0; ±5% Gaussian noise in process output.

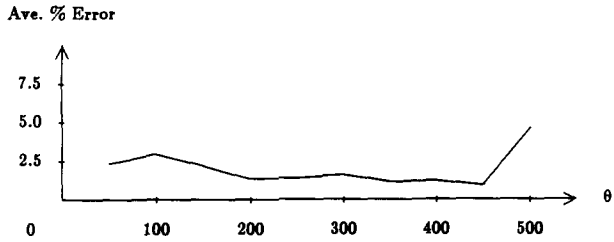

Figure 4. Error in delay identification as a function of θ. τ_p = 100 sec; K_p = 1.0; ±5% Gaussian noise in process output.

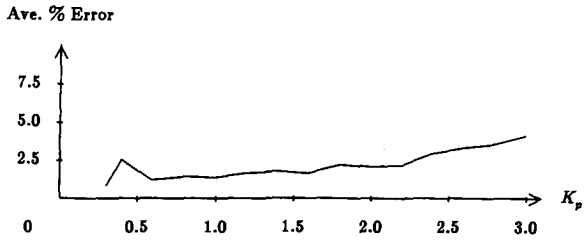

Figure 5. Error in delay identification as a function of K_p. θ (actual) = 250 sec; τ_p = 100 sec; ±5% Gaussian noise in process output.

ter. The remaining parameters were held constant at or near the midpoints of their training ranges. Based on these results, the following observations can be made:

- Overall, the average error of estimation is less than 7 sec, or 1.4% of the range of delay values.
- The average percent error is within 2.5% over a wide range of delays and process time constants. In absolute terms, the average error is between 2 and 5 sec except in the extremes of the delay range.
- For parameter values within training ranges, estimation error is small. There is one major exception. For very small delays, percentage error is large. This is to be expected. The sampled process output in this case provides little relevant information. It is likely that a nonuniform sampling rate would overcome this problem.
- In many cases, estimation error is acceptable even for parameter values outside training ranges. For example, the error for $\tau_p = 280$ is less than 4%. Even for gains twice as high as any the network was trained on, the error is around 4%.
- Estimation is robust with respect to noise. For 25% noise, the average error is about 6.5%. For 50% noise, the average error reaches 13%.

In some applications, a noise level of 5% (of the range of process response to the step input) cannot be assumed. As a stiffer performance test, we separately trained a network with $\pm 50\%$ noise (all other training parameters were identical to the above experiment). The average error after training in this case was less than 9%, compared to 13% for the network trained with 5% noise. The noise sensitivities of these two networks makes for an illuminating comparison (Figure 6). At low noise levels,the network trained with 5% noise performs significantly better. At higher noise levels, the inequality is reversed.

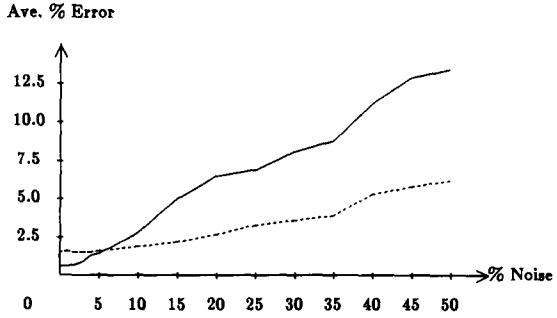

Figure 6. Error in delay identification as a function of noise for networks trained with 5% noise (solid line) and 50% noise (dashed line). θ (actual) = 250 sec; $\tau_p = 100$ sec; $K_p = 1.0$.

4.2. Using the Delay Identifier

Once the network has been trained, it can be used for online delay identification. This operational phase is relatively straightforward. The input to the network is now actual (not simulated) process output. The output of the network is again a delay estimate. This delay estimate can then be used for control. For example, if the process controller incorporates a Smith predictor or other delay compensation technique (Stephanopoulos [10]), the delay estimate can be given as input to it. The Smith predictor consists of a process model that is exercised with and without the delay. The difference between the nondelayed process output estimate and the delayed estimate is added to the measured output before being input to the controller, which is otherwise unchanged.

Figure 7 depicts this application of the delay identifier. When a delay estimate is needed, the control loop is broken (switch S1) and a (small) step input perturbation is applied to the process. The response of the process to the perturbation is sampled and stored in a buffer. When a sufficient number of samples have been received, the vector of samples (scaled appropriately) is used as input to the trained neural network. The output of the network is subjected to some postprocessing (scaling and/or translation) to obtain a delay estimate θ_{est}. Once the delay estimate has

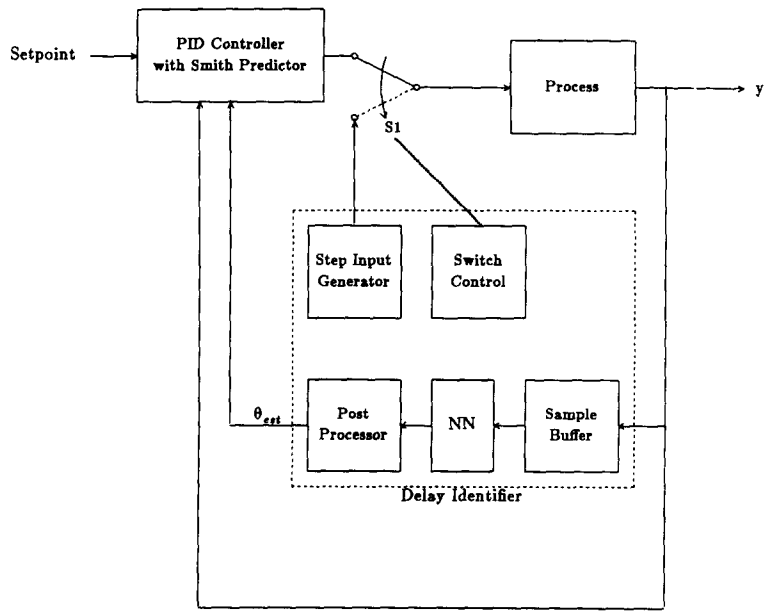

Figure 7. Using the neural network delay identifier.

been input to the Smith predictor, S1 can be closed again and the process put back under closed-loop control.

We have simulated the setup of Figure 7 and investigated the effect of delay identification on closed-loop control quality. A first-order process and a simple proportional controller, with the controller gain set to unity, were used for the simulation (Figure 8). Without knowledge of the process delay, fast response of the closed-loop system can be achieved only at the expense of considerable oscillation. With a Smith predictor and the delay-identifier-supplied estimate, significantly better control is achieved: the process response attains steady state rapidly and without overshoot.

5. SOME EXTENSIONS

For our proof-of-concept system we simplified a number of aspects of system identification in general. Here we briefly discuss extensions that can help overcome some of these simplifications. Although the discussion in this section is in terms of delay identification, the schemes described can be adapted for many other system identification problems.

5.1. Recursive Estimation

A distinction is often made between "recursive" and "batch" system identification techniques. Recursive techniques operate as data are received. Although the accuracy of the estimates will increase with more data, the preliminary estimates can often be sufficiently accurate. In batch estimation, a predetermined number of samples are collected before an estimate is produced.

As presented above, our approach can be used for developing batch identifiers. However, there are at least three extensions of the basic approach that can facilitate recursive identification. First, the missing samples can be extrapolated from the observed data. There are a number of ways this can be done. One technique we have briefly investigated is the following. The unknown samples can be generated by using the process model with "likely" parameter values. For example, the parameter values from the last estimate can be used. In our delay identifier, we have used parameter values in the centers of their training ranges. Results have been positive. In almost every case, an estimate within $\pm 10\%$ is produced almost immediately after the delay and is then refined further as more samples are processed.

Second, an interpolation approach is possible. If x samples are available and 50 are required,the time scale can be multiplied by a factor of $50/x$. Fifty sample values can be generated by interpolating between true samples, using the last true sample as the 50th. The 50 samples are then used

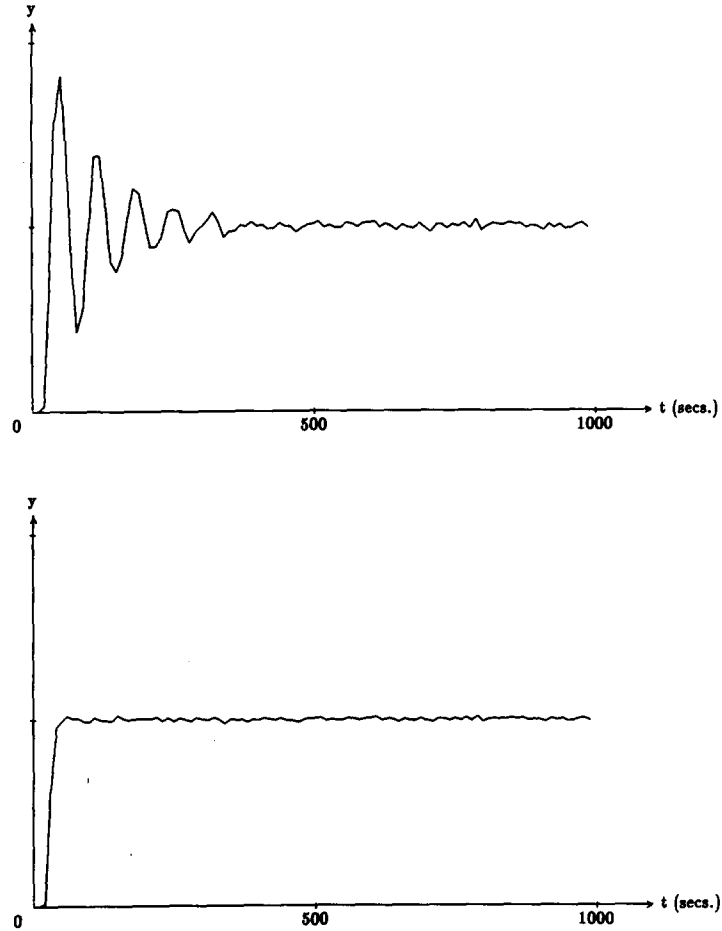

Figure 8. The benefit of delay identification. Closed-loop responses of a first-order process to a step change in setpoint. The top curve shows the response when the controller is not provided a delay estimate. The bottom curve shows that response is much better with a good delay estimate. τ_p = 10.0 sec; K_p = 1.0; θ (actual) = 25.0 sec; θ_{est} = 24.0 sec (for bottom curve). The controller gain was unity for both curves.

to produce a delay estimate, which is then multiplied by $x/50$ to renormalize the time scale.

A third approach is similar to the above except that all the needed samples are collected within the smallest time window in which identification will be required. For identification based on a larger time window that is an integral multiple of the smallest, some samples are not used. This approach can be combined with the interpolation approach. The process

can be sampled at a higher rate than needed for batch estimation, and interpolation can be used where needed.

5.2. Toward Closed-Loop Estimation

The essence of our approach is the approximation of a function from process input/output to parameter value estimates. In the open-loop case, the process input can be a constant and therefore need not be provided as input to the network. For general closed-loop identification, estimates have to be produced given continuously varying inputs. In principle, there appears to be no reason why a network could not be trained for this case as well. However, the generation of training examples would be more complicated, and we can expect that many more training examples will be needed.

The general closed-loop extension of our approach is a research effort in itself. In this project, we have investigated a constrained form of closed-loop identification: delay identification under "bang-bang" control. In closed-loop bang-bang control, the process can be switched on or off. Whenever the output exceeds an upper bound, the process is turned off; whenever the output falls below a lower bound, the process is turned on. Bang-bang control is commonly used in HVAC (heating, ventilation, and cooling) systems.

For delay identification under bang-bang control, we assume that the collection of output samples is initiated when the process is turned on. After the predetermined number of samples have been collected, and provided that the process has not been turned off in the interim (this restriction can be relaxed fairly easily), an estimate is produced. Given this scenario, there is only one difference between open-loop and bang-bang delay identification. In the former case, the process is assumed to be at a constant value (except for noise) from when the step input is given until the delay expires; in the bang-bang case, the process output is decaying during the delay. The decaying and rising responses can be governed by different dynamics.

We have trained a network to identify the delay of a process under bang-bang control. It was assumed that both the "on" process and the "off" process were first-order with independent (and therefore different) time constants. The process input was again constant over the duration of a training example and was not provided to the network. An average error rate of around 7% was achieved in 100,000 iterations. The network converges significantly faster than for the open-loop delay identification, and we expect that a comparably long simulation would produce lower error rates. The better performance in the bang-bang closed-loop case is not too surprising—a transition between a falling and a rising curve is easier to detect than a transition between a constant and a rising curve.

5.3. Application to Higher Order Processes

The process models generally assumed for purposes of control and diagnosis are simplifications of the actual process. It is thus important to see how an identification technique performs when tested on processes more complex than the model that is the target of the identification. For example, it is important that a delay identifier for first-order processes produce reasonable estimates when used for a higher order process. The definition of "reasonable" in this case is somewhat murky. For example, the response of a first-order process with a given delay and time constant can be very different from that of a second-order process with the same delay and time constant. Functionally, the estimates should allow the process to be controlled or diagnosed adequately.

We have evaluated our delay identifier for second-order processes as follows. A second-order process model (with delay) is used to generate a process step response. The sampled response is given as input to the (first-order) delay identifier, which produces an estimate of the "equivalent" first-order process delay. First-order curves using the estimated delay are generated by varying the values of other parameters to see if the second-order response fits some first-order curve with the estimated delay. By this criterion, our proof-of-concept system performs very well. A more stringent test would be to develop estimators for all first-order parameters and generate the corresponding curve to observe the fit. This was beyond the scope of the effort.

6. ESTIMATION OF OTHER MODEL PARAMETERS

Although most of our work on parameter estimation with neural networks has focused on delay identification for first-order processes, we conducted preliminary experiments for estimating other model parameters. Some of our results are indicated below. Unless otherwise indicated, normalized gains in the process simulation ranged from 0.5 to 1.5, time constants from 20 to 180 sec, damping coefficients for underdamped processes from 0.4 to 1.0 (exclusive) and for overdamped processes from 1.0 (exclusive) to 6.0, and the noise level was $\pm 50\%$. We reiterate that the ranges for temporal parameters can be scaled by changing the sampling period. Training was not as extensive as for our first-order delay identification networks.

- For the time constant of second-order underdamped processes (without delay) we achieved an average error of 6.2 sec. Thus the error was 3.9% of the 20-180 sec range.

- For the delay of second-order delayed underdamped processes ranging from 0 to 400 sec, we achieved an average error of 16 sec (4%).
- For delay of second-order delayed overdamped processes, the average error was 25 sec(or 6.3% of the range of delay values, 0–400 sec). The time constant range was 40–160 sec.

7. CONCLUSIONS

Nonlinear approximation techniques are often needed in linear systems. In this paper, we have discussed an important nonlinear problem—identification of process model parameters. Neural networks provide an attractive technology for solving this problem. They do not require the control theory and process engineering expertise that many conventional system identification techniques demand, they exhibit high generalization and noise tolerance, and their online implementation is compact and fast. Much of this paper has focused on a specific application—open-loop delay estimation. High accuracies have been obtained, and the generalization and robustness properties of neural networks have been verified. The approach is amenable to a variety of system identification applications, although for the more demanding ones, such as closed-loop control of multivariable systems, significant extensions are likely to be needed.

Finally, we briefly mention an alternative approach to parameter estimation for transfer function models. As the point of parameter estimation is to achieve a good fit between actual process data and the process model response, nonlinear optimization algorithms can be applied online to minimize the estimation error. This approach does not require the extensive offline training phase of a pure neural network approach and will likely have higher accuracy. On the other hand, it is more demanding, in terms of computation and memory, of the online implementation. The complementary strengths and weaknesses of the neural and algorithmic approaches are an indication that a hybrid system might be appropriate. We are currently investigating hybrid approaches to system identification that promise the best of both worlds (Konar and Samad [11]).

References

1. Barto, A. G., Connectionist learning for control: an overview, in *Neural Networks for Control* (W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds.), MIT Press, Cambridge, Mass., 1990, pp. 5–58.

2. Samad, T., Neural networks for control—an overview, *Proceedings ISA/91*, Anaheim, 1991.

3. Bhat, N., and McAvoy, T., Use of neural nets for dynamic modeling and control of chemical process systems, *Proceedings of the American Control Conference*, Pittsburgh, 1342–1348, 1989.

4. Narendra, K. S., and Parthasarathy, K., Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Networks* **1**(1), 4–27, 1990.

5. Samad, T., and Mathur, A., Parameter estimation for process control with neural networks, in *Applications of Artificial Neural Networks*, vol. II (S. K. Rogers, Ed.), Proc. SPIE 1469, 766–777, 1991.

6. Ljung, L., *System Identification: Theory for the User*, Prentice-Hall, Englewood Cliffs, N.J., 1987.

7. Werbos, P., Beyond regression: new tools for prediction and analysis in the behavioral sciences, Ph.D. Thesis, Harvard Univ. Committee on Applied Mathematics, 1974.

8. Rumelhart, D. E., Hinton, G. E., and Williams, R. J., Learning Internal Representations by Error-Propagation, ICS Rep. 8506, Institute for Cognitive Science, UCSD, La Jolla, Calif., 1985.

9. Hornik, K., Stinchombe, M., and White, H., Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, *Neural Networks* **3**, 551–560, 1990.

10. Stephanopoulos, G., *Chemical Process Control: An Introduction to Theory and Practice*, Prentice-Hall, Englewood Cliffs, N.J., 1984.

11. Konar, A. F., and Samad, T., Hybrid Neural-Network/Algorithmic System Identification, Tech. Rep. SSDC-91-I4052-1, Honeywell SSDC, Minneapolis, 1991.