

Using a neural network system for advanced process control

John B. Rudd

Radically different from common computing techniques, neural networks are massively parallel systems that rely on dense arrangements of interconnections and surprisingly simple processors.

Two problems that have always plagued the process control engineer are: (a) lengthy delays in achieving the intended outcome in making adjustments, and (b) a need to control variables that are not directly or instantaneously measurable. These variables are sampled physically and measured in the laboratory on a periodic basis, such as once an hour or once per shift. Adjustments are then made to the process variables that indirectly and belatedly result in the desired effects on the control variables.

With the advent of electronic instrumentation and microprocessor-based control systems, attempts at more direct control have been developed using mathematical modeling and simulation. Often, these simulations are limited by simplifying assumptions made to achieve some degree of linearity between adjustments and process results. Such a simplified model can be used for direct control, as long as the operators understand the limitations and the controls are operating within the range for which the model was designed.

To overcome these kinds of limitations, we have to develop a full model based on scientific first principles. To do so requires an extensive understanding of the process and the capability to reduce that understanding to a comprehensive mathematical representation. First principle models have proven effective in operation. However, the complex mathematical models have to be maintained, which demands the same capability and understanding as needed to develop the original model.

An alternative solution to these types of problems is the artificial neural network. These networks establish a computing architecture that has unusual and unique capabilities (1-3). Artificial neural network architectures are made up of elements that perform in a manner analogous to the basic biological neuron, or nerve cell. The artificial neurons are organized in a manner that may or may not simulate the anatomy of the brain.

Despite the somewhat superficial resemblance to the biological model, artificial neural networks exhibit a number of the brain's characteristics. Three similarities

that make neural networks suitable for use in process control are: (a) the ability to learn from experience, (b) the ability to generalize from previous examples to new ones, and (c) the ability to extract essential characteristics from inputs containing irrelevant information.

Background on neural network technology

Figure 1 shows both a single biological neuron and a simplified network with nerve cells. The set of inward-branching fibers called "dendrites" may be thought of as making up the input area of the cell. The dendrites are connected to the cell body, the "soma." The long fiber denoted as the "axon" is, by analogy, the output area of the cell and is connected to a set of outward-branching fibers called "boutons."

As the soma receives stimuli through its dendrites, its activity level increases until a threshold is reached. Then the cell emits an impulse that travels along the axon and its branches to the ends of the boutons. The connection points from boutons to other cell dendrites or to somas are known as "synapses."

Figure 2 shows the neuron model used to simulate these characteristics of the biological neuron. Inputs represent either an input from outside the network or the output of another neuron within the network. Each input is multiplied by a weight factor, which is analogous to the synaptic strength of the biological neuron. The weighted inputs are then summed to determine the activation level of the neuron. The output can be represented mathematically as:

$$Y = f(X_1 W_1 + X_2 W_2 + \dots + X_N W_N) \quad (1A)$$

or

$$Y = f\left(\sum_{n=1}^N X_n W_n\right) \quad (1B)$$

where

Y = output from a neuron

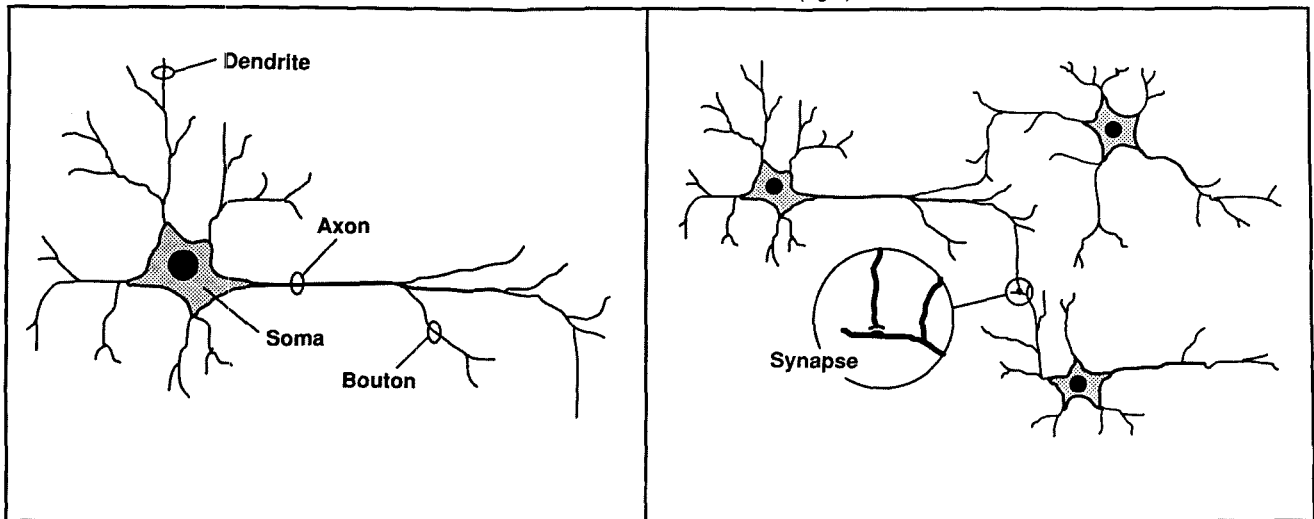
X_n = n th input to a neuron

W_n = n th weight applied to n th input.

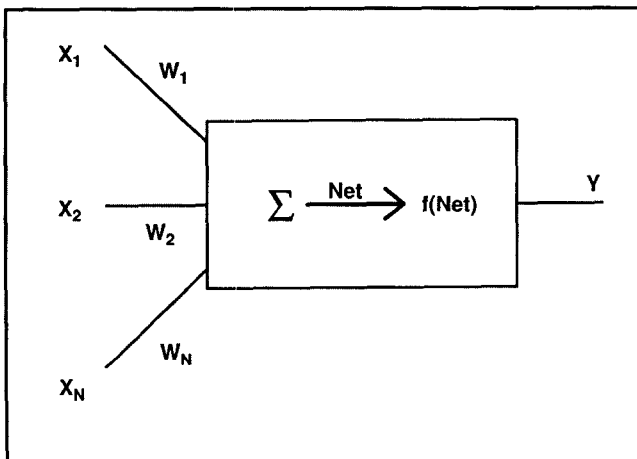
The function is normally referred to as a "squashing"

Rudd is manager of the Advanced Application Technology Group, Automation Technology, Inc., Mobile, AL 36691

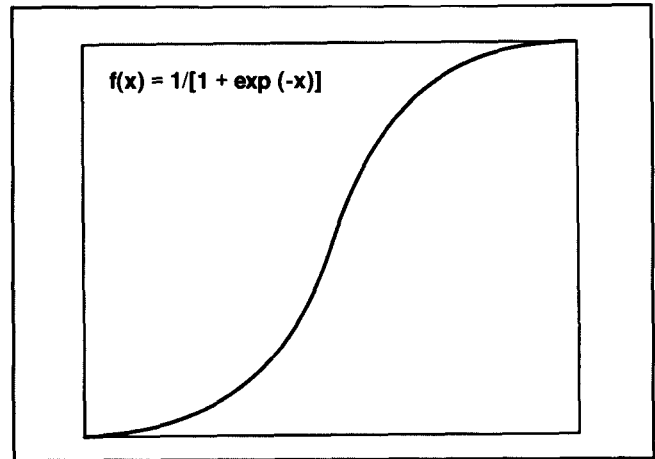
1. A single biological neuron (left) and a simplified network of a nerve cell in the brain (right)



2. Artificial neuron model



3. General sigmoid function



function, which can be any bounded, monotonic, increasing function. The use of this function adds nonlinearity and stability to the network. One of the most commonly used squashing functions is the sigmoid function, which is represented in Fig. 3. This function has the general form:

$$f(x) = 1 / [1 + \exp(-x)] \quad (2)$$

This fundamental processing element, the neuron or node, is the basic unit on which neural networks are built.

Back-propagation

There are many network paradigms in use today. For the multivariable continuous processes in pulp and paper mills, one successful network paradigm is the back-propagation network. It is one of the easiest networks to understand because its learning and update procedures are intuitively appealing. They are based on a relatively simple concept.

Back-propagation was first introduced in 1974 by Paul Werbos and was independently rediscovered by David Parker in 1982. During the mid-1980s, the capabilities and

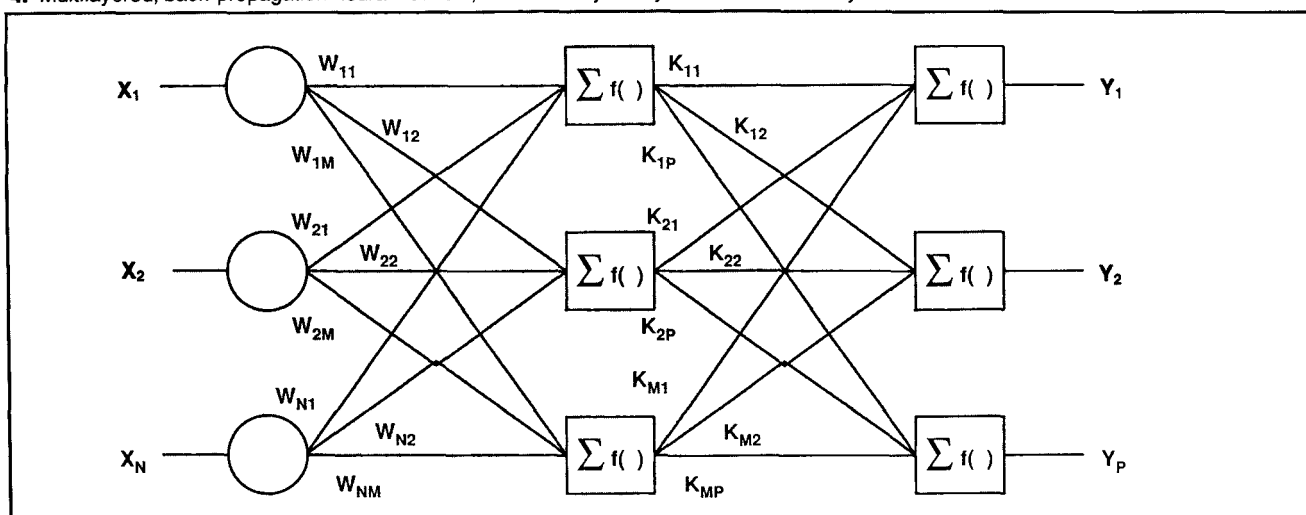
speeds of computers increased, and the costs fell sharply. The proliferation of computers provided a fertile environment for the third independent reinvention of the concept in 1986 (4), which sparked widespread interest in neural network simulations.

Architecture

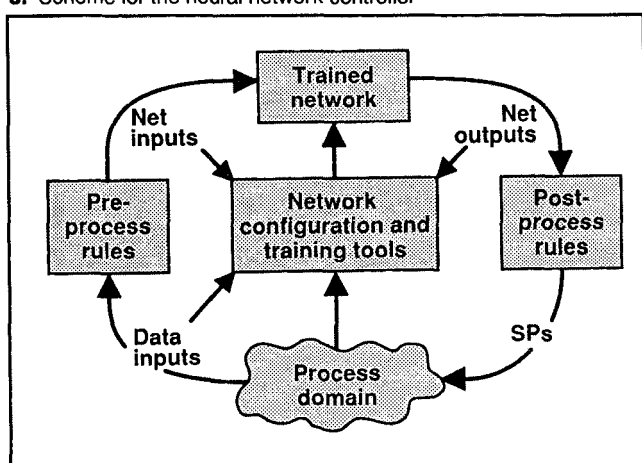
Back-propagation networks are usually layered, as shown in Fig. 4, with each layer being fully connected to the layers before and after. In operation, the network is supplied with a group of inputs. The updating of activation values propagates forward from the input neurons, through the hidden layer of neurons, to the output neurons that provide the network response.

When the network is training, the results of the forward pass through the network are compared to known results, and a standard error is calculated to be used for correcting the network weights. The correction mechanism starts with the output neurons and propagates back in the reverse direction—hence the term “back-error propagation,” or simply “back-propagation.” This process is continued until the network predictions are within some defined tolerance of acceptability.

4. Multilayered, back-propagation neural network, with each layer fully connected to the layers before and after



5. Scheme for the neural network controller



Let us look at the biological learning process. Usually, certain conditions are set up, and an expected action or response is demonstrated. For example, a child is shown a card on which the letters "c a t" are printed. The child is instructed that these letters represent the name of an animal, and the animal is a cat.

In another lesson, the child may be shown cards with letters like "c o w," "d o g," or "h o r s e." The cards are shown repeatedly, and the child is asked to respond with the name of the animal. At first, the child will guess incorrectly but will soon begin to recognize the names. After some number of repetitions, the child will respond quickly and correctly with the right names.

This is the same basic technique used to train the artificial neural network. If the network gives the wrong answer, the weights are corrected, with the result that the error is reduced and further responses are more likely to be correct.

Referring again to the multilayered network in Fig. 4, the outputs can be mathematically represented by:

$$Y_p = f\left(\sum_{m=1}^M \left[f\left(\sum_{n=1}^N X_n W_{nm}\right) K_{mp}\right]\right) \quad (3)$$

where

Y_p = the p th output of the network

K_{mp} = the p th weighting factor applied to the m th output of the hidden layer

W_{nm} = the m th weighting factor applied to the n th input to the network

X_n = the n th input to the network.

Advantages and qualifications

Some of the real advantages of using a neural network in a process control application are:

1. Neural networks are not programmed. They learn by example with no requirement of prior knowledge.
2. Neural networks are robust in nature and can operate even if portions of the input data are incorrect. The network can see through noise and distortions to obtain the true essence of real-world environment being viewed.
3. Neural networks can make generalizations to reach new conclusions based on past experience.
4. Neural networks can solve any problem involving the mapping of input data into output data.

As with most useful concepts, however, these advantages come with qualifications that must be evaluated. To begin with, learning takes time. Depending on the complexity of the network and the quantity of classification data (i.e., training sets), the learning process can take hours, days, or even weeks. Moreover, even though the networks can see through noise and distortions most of the time, there may be times when the network will be tricked and will "see" its equivalent of an optical illusion.

Being good at making generalizations and reaching new conclusions does not lead to being precise or logical. Consider the following problem:

$$2.14325 + 3.25617 = 5.39942$$

The neural network would conclude that adding the two numbers together will result in a number that is probably very close to 5.4.

To take a logical problem, let us consider the simple relationship:

$$\text{If } a = b \text{ and } b = c, \text{ then } a = c.$$

The neural network would conclude that the term a is equal to something, and it's probably really close to the term c .

When neural networks are properly applied, these qualifications are not really problems, especially when the meaning and implications of the qualifications are understood and are considered in managing the controls.

Understanding the results and limitations

Structure and training

One of the main advantages of the neural network is that it learns from example without any need for prior knowledge. By contrast, a rule-based expert system requires an exact knowledge base that relies on rules that humans must supply. As pointed out in a study regarding the modeling of human operators using neural networks (5), it can be difficult to obtain the required rules for the rule-based system. Many factors related to human nature must be considered to develop a complete set of rules. Also, the number of rules required can be extensive because there must be a rule for every case.

On the other hand, the knowledge base is relatively simple to obtain. All that is required are actual responses based on actual inputs. Since the neural network learns and makes generalizations based on the actual data, relatively few training sets will produce accurate results.

To illustrate, a neural network was developed to classify music. The network had 25 inputs representing the pitch and duration of the first 12 notes of a song, along with the tempo of the song. The output simply gave the class of music in which a particular song belonged: classical, hymn, rock, and garbage. Eight songs were used to train the network, and the network was asked to classify 12 songs.

In all 12 cases, the proper selection was made with better than 57% assurance. The worst-case example was encountered when the network was asked to classify the hymn "Rock of Ages." The network responded with a 57% chance it was a hymn, a 32% chance it was rock, and an 11% chance it was garbage.

The 12 test cases were combined with the original eight training cases, and the network was retrained. Presented with 10 new songs to classify, the network classified all 10 with better than 96% assurance.

An important fact is that this network was developed and trained by individuals who could read music but could not compose music. For them to have developed a rule-based system would have been a prohibitive task. It would have meant defining and describing a mathematical theory of organizing tones in terms of melody, harmony, and rhythm. It might be possible to construct a rule-based system that would return the same results as the neural network, but the knowledge base would be extensive, and the rules would be highly complex.

Time needed for a neural network to learn

The music classification network was trained in less than an hour on a personal computer equipped with an Intel 80386 microprocessor and an Intel 80387 math coprocessor.

Other networks that are more complex have taken two to three days to train. One mathematically generated model with 52 inputs, 30 hidden nodes, 10 outputs, and 1500 training sets did not complete training after 20 days.

The training of the network is affected by the number of inputs and outputs, the number of hidden layers and hidden nodes, and the desired accuracy. The training technique, the speed of the computer, the mathematical tools used to evaluate the results, and the number of learning thresholds evaluated also affect the time required for learning and evaluation.

Inputs and outputs. Just as with any control problem, the inputs and outputs along with their interrelationships and dependencies must be considered in planning the network. With today's distributed control systems, a wealth of information is available. In many cases, it is tempting to simply use everything, but the secret to a good neural network is to fit the data to the problem without overfitting.

The training process creates a statistical map relating inputs to outputs. Having inputs that do not relate to an output forces the network to determine that there is no relationship and to set the weights for the nonrelated input to zero. When the network is operating, it has to perform needless computations in multiplying these inputs by zero.

Another case of overfitting is to have two inputs that represent the same basic information. For example, if both the process variable of a flow loop and the valve position of the flow control valve are used as inputs to the network, the effect is that of a single input having been entered twice. The network uses normalized data, so when both signals are converted to relative values between 0 and 1, both the valve position and flow measurement represent the same basic data. Not only is this needless computation, but it tends to bias the network in regard to that signal.

Hidden nodes. The number of hidden layers and hidden nodes is important because of the number of calculations that must be made in both the training and operational modes (as indicated by Eq. 3). How many hidden layers produce the best results? In general, for feedforward, back-propagation networks, a network with multiple hidden layers can be replaced by an equivalent network with a single hidden layer (4).

In the single hidden layer, then, how many hidden nodes are required? One way to determine the number of nodes is to train the network using different numbers of hidden nodes, testing each permutation for accuracy of prediction.

Training and accuracy. In one case at our company, a 32-input, 3-output network was developed and trained over a number of permutations with hidden nodes incremented in five-node steps from 5 to 60 nodes. (Each permutation can be considered a different network, because each has to be independently trained.)

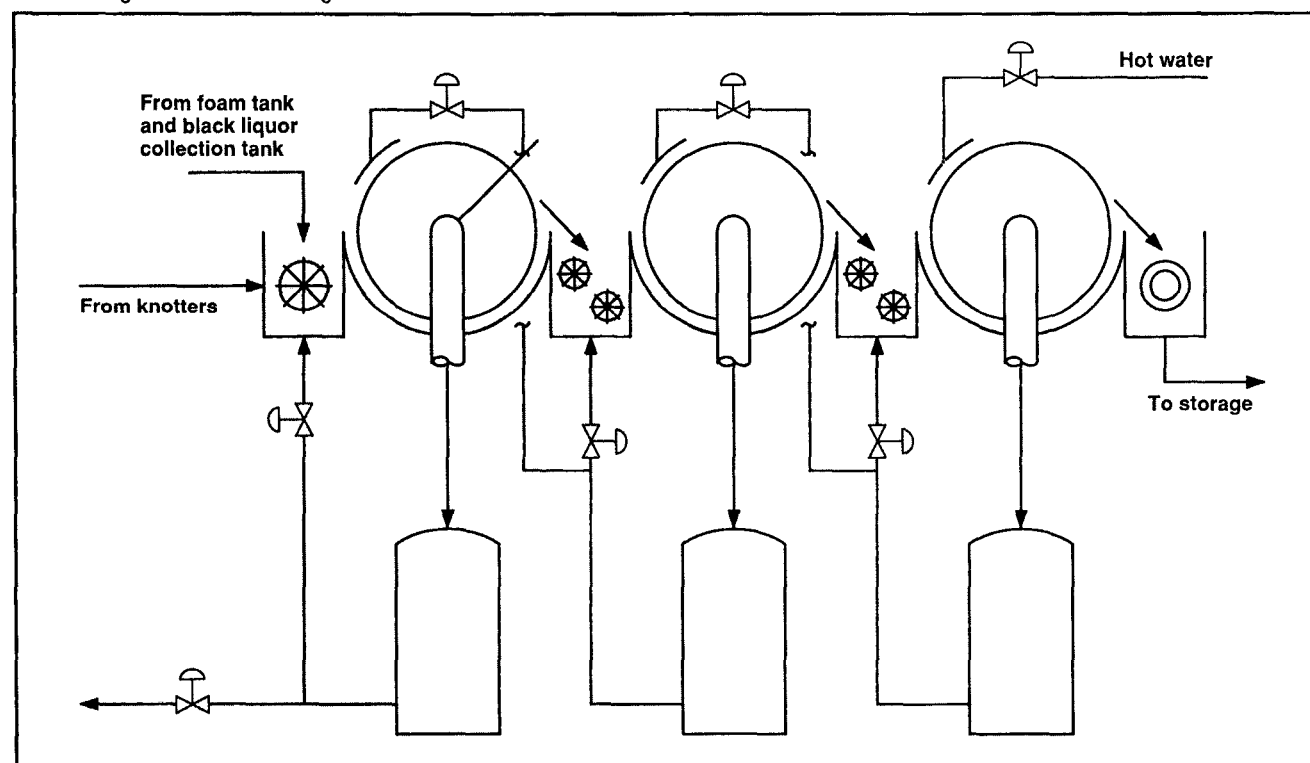
The mathematical tools used to evaluate the results were:

1. The overall average root mean square (RMS) with the range of average RMS values per sample for all outputs combined
2. The individual average RMS with sample ranges for each output individually
3. The percentage of the total number of outputs exceeding a percent error of 7.5%
4. The percentage of test sets with at least one output exceeding a percent error of 7.5%

1. Comparison of results from the neural network, regression models, and laboratory samples for the bleach washer

	Rate, tons/day				Mean
	650	650	850	850	
	Consistency, %				
Actual sample	12.60	12.40	11.80	11.20	12.00
Neural network	12.52	12.42	11.72	11.28	11.99
Regression models					
Simple	12.80	12.80	11.48	11.48	12.14
Multivariable	12.38	12.61	11.58	11.44	12.00
	Percent error, %				
Neural network	8	2	8	8	6.50
Regression models					
Simple	20	40	32	28	30.00
Multivariable	22	21	22	24	22.25

6. Flow diagram for the three-stage brownstock washer



5. The percentage of each individual output exceeding a percent error of 7.5%.

Using all of these tools together, we found that two levels of hidden nodes provided similar results: 25 and 40 hidden nodes. Since the results were approximately the same, we chose the 25-node networks for further evaluation because the computing time is less for training and for operation. The 25-node networks were trained with 22, 23, 24, 25, and 26 hidden nodes, and the same type of training analysis was performed. No efforts were made to train networks past 26 nodes because we noticed an obvious deterioration in accuracy.

Learning thresholds. These networks were trained and evaluated at five different learning thresholds. These five

thresholds had provided the best results with the initial networks: 0.0081, 0.0064, 0.005, 0.0036, and 0.0025. The particular learning threshold used is defined as:

$$\text{Threshold} = 0.8 \times (\sum \epsilon^2 / 2) \quad (4)$$

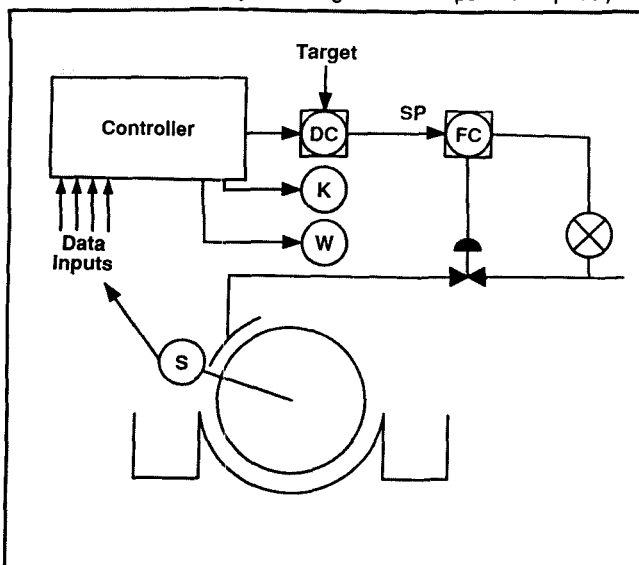
where

ϵ = normalized error between the network output and the actual measured output.

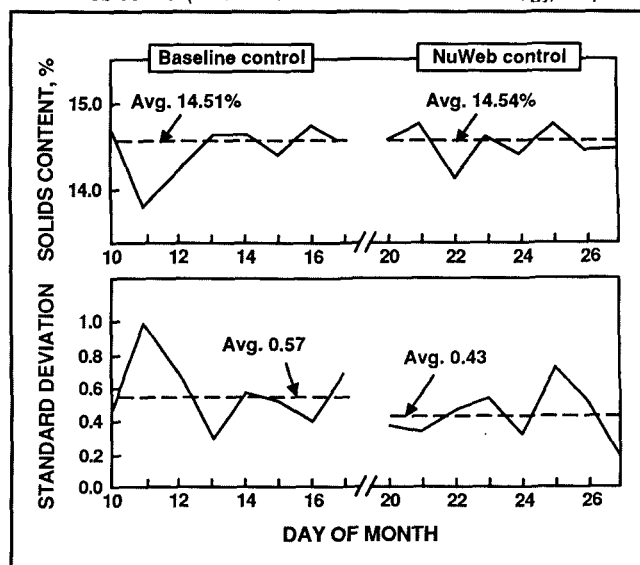
Using this analysis, the best network was found to have 25 hidden nodes operating with a learning threshold of 0.0064.

Improving training times. Much can be done to

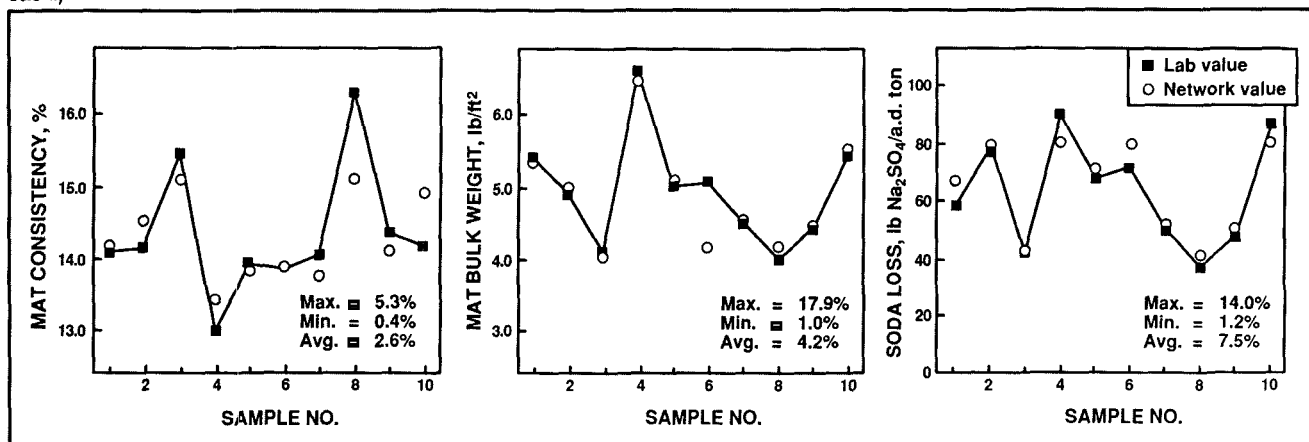
7. Washer control strategy (DC = dilution factor controller. FC = flow controller. K = consistency. W = weight. SP = set point. S = speed.)



8. Reduction in the standard deviation of the filtrate solids by using the NuWeb control (a trademark of Automation Technology, Inc.)



9. Test results for consistency, mat bulk weight, and soda loss. (Values for maximum, minimum, and average percent error are given for each.)



improve training times (6). One of the best ways to cut training time is to use a hardware accelerator card. These circuit boards are capable of processing networks 50–100 times faster than a 25-MHz 80386/80387 personal computer. With these special-purpose cards, networks that might require a day of training would take a matter of minutes to train.

The neural network controller

As stated, there are many problems in pulp and paper mills that make control difficult. A hybrid system that combines a neural network system, a rule-based system, and a conventional computational system may be needed to handle these problems with operational simplicity and effectiveness.

Figure 5 illustrates the concept used in a neural network controller. Data are collected from the process domain along with known results to train a configured neural network. In training the network, process information is collected from the process domain control system

and is passed through a rule-based preprocessing element before going to the network.

Once the network is run, the results, in process variables and controller set points, are passed to the postprocessing element. The preprocessing and postprocessing systems are a combination of mathematical calculations and forward chaining rules that operate on current input data, historized data, and current network outputs to provide a stability the neural network does not have on its own.

Our first effort to determine the ability of a neural network to provide meaningful results was based on a bleached washer. One variable that has always been missing when trying to perform mass balances around washers is the consistency of the mat leaving the washer. We developed a network that viewed the world around the washer as represented by 14 variables available in an existing control system.

Sets of data samples were collected at four positions across the face of the washer at production rates of 500, 600, 700, 800, and 900 tons/day. The consistency of each sample was determined, and the four sample values within

each set were averaged to determine the overall mat consistency. These data were used to train the network to output a value for the mat consistency. The network computation was executed every 5 s to simulate a sensor process variable.

The process was run at 650 and 850 tons/day, and additional data sets were collected to be used for testing the trained network. In the past, we have used regression analysis to develop models that could be implemented within the structure of a distributed control system. These models simply correlated the production rate to the mat consistency. This same exercise was carried out using the collected data. In addition, a multivariable regression model was developed using all the variables used by the neural network. The test data were used to evaluate both regression models and the neural network.

Table I shows the results of the testing expressed in terms of percent error between measured and predicted results. After seeing these results, we developed a dynamic simulation of the process and implemented a neural network controller to manipulate certain process variables related to the washer operation. Our effort was directed at controlling the consistency of the mat leaving the simulated bleach washer. Based on the success of this simulation, we felt that a full mill trial was justified.

The mill trial

For the mill trial, we chose a brownstock washer as the subject for control. The preliminary results of the trial were reported previously (7). The brownstock washer line used for the trial was a three-stage, 850-ton/day, hardwood line.

Figure 6 presents the flow diagram. The washer feed is delivered from two pressure knotters with liquor from a foam tank and from a liquor collection tank, reintroducing liquor collected from the washer system back into the first washer vat. Forty-four variables were considered as possible points to become part of a training set. Every effort was made to take into account all aspects of the process that could possibly affect the washer operation.

A testing program was set up to collect laboratory data to be used for training the neural network. As with the bleach washer example, four samples were taken across the face of the washer. Laboratory testing was carried out on each sample to determine unit mass in lb/ft², mat consistency in percent, and soda loss in lb Na₂SO₄/a.d. ton.

The neural network controller was initially implemented to provide software sensor values for mat consistency and mat density and to control the dilution factor to stabilize the black liquor solids. Figure 7 illustrates the strategy. This control scheme was implemented, and the result was a 25% reduction in the standard deviation of the black liquor solids, based on an eight-day trial. The controller was on control for the entire eight days, as shown in Fig. 8.

Twice during the trial, one of the two knotters feeding the washer line went down, resulting in 15–20% of the input variables going out of range. In both instances, the controller continued to make valid predictions, and the dilution factor controller maintained automatic control.

At the completion of the initial trial, operations at the mill changed, resulting in a modification of the operating range of the washer line. When the network was required

to operate outside the original training domain, the results began to deteriorate. More data were collected to expand the range of the process domain viewed by the controller, and the network was trained to predict soda loss values in addition to consistency and mat density.

Figure 9 shows the test results of samples taken to test the network. The controls of consistency and mat density continue to perform with the same accuracy as during the previous trial. The software sensor value for soda loss is also being predicted at an acceptable level. All of these variables are being predicted every 15 s. This timing is sufficient for use as sensor PVs in control strategies.

Conclusions

The kinds of variables we have dealt with here are those for which values have always been difficult to obtain. Values have been obtained in laboratory testing, or they have been obtained from sampling instruments through an extended time cycle. Some of the variables are meant to replace on-line analytical sensors that are high maintenance items with imperfect results.

As an alternative to traditional modeling techniques for obtaining these variables, neural networks, rule-based systems, and conventional mathematical techniques can be combined. Software sensors can be created to provide values for the variables.

In the proposed system, empirical models are based on data and principles of statistics, not on a need for fundamental insights as to the working relationships in the process. The models are complex, but they are trained on data. There is minimum fundamental theory, and there are no software bottlenecks.

The models are easy to understand, since the quantification of all variables is represented in a map relating inputs to outputs. For nonlinear systems, the neural network models work well, providing clues as to underlying physical phenomena.

With the addition of these new, reliable process variables, the control engineer can develop controls that have been marginal in the past. Many supervisory controls that were previously unattainable are now opportunities for the industrious control engineer.

Literature cited

1. Wasserman, P. D., *Neural Computing, Theory and Practice*, Van Nostrand Reinhold, New York, 1989.
2. Dayhoff, J., *Network Architectures: An Introduction*, Van Nostrand Reinhold, New York, 1990.
3. Minor, J. M., Proceedings of the ISA 1990 International Conference and Exhibition, Vol. 3, Instrument Soc. Am., Research Triangle Park, N.C., p. 1467.
4. Ramelhart, D. E. and McClelland, J. L., *Parallel Distributed Processing*, Vol. 2, MIT Press, Cambridge, Mass., 1986.
5. Gingrich, C. G., Kuespert, D. R., and McAvoy, T. J., Proceedings of the ISA 1990 International Conference and Exhibition, Vol. 3, Instrument Soc. Am., Research Triangle Park, N.C., p. 1481.
6. Caudell, M., *AI Expert* 6(1): 56(1991).
7. DeGroot, D. L. and Rudd, J. B., 1991 TAPPI Process Control Conference Proceedings, TAPPI PRESS, Atlanta, p. 119.

Received for review May 30, 1991.

Accepted July 10, 1991.

Presented at the TAPPI 1991 Engineering Conference.