Conf-920538--14

# A UNIVERSAL, FAULT-TOLERANT, NON-LINEAR ANALYTIC NETWORK FOR MODELING AND FAULT DETECTION*

ANL/CP--75747

DE92 011840

by
J. E. Mott, R. W. King, L. R. Monson,
D. L. Olson, and J. D. Staffon

Argonne National Laboratory
Idaho Falls, ID

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

The submitted manuscript has been authored by a contractor of the U. S. Government under contract No. W-31-109-ENG-38. Accordingly, the U. S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U. S. Government purposes.

*Submitted for Presentation at the*
*8th Power Plant Dynamics, Control & Testing Symposium*
*University of Tennessee*
*May 27-29, 1992*

March 6, 1992

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

# A UNIVERSAL, FAULT-TOLERANT, NON-LINEAR ANALYTIC NETWORK FOR MODELING AND FAULT DETECTION

J. E. Mott
Advanced Modeling Techniques Corporation
1820 East 17th Street, Suite 375
Idaho Falls, ID 83404

R. W. King, L. R. Monson, D. L. Olson, and J. D. Staffon
Argonne National Laboratory - West
P.O. Box 2528
Idaho Falls, ID 83403-2528

## ABSTRACT

The similarities and differences of a universal network to normal neural networks are outlined. The description and application of a universal network is discussed by showing how a simple linear system is modeled by normal techniques and by universal network techniques. A full implementation of the universal network as universal process modeling software on a dedicated computer system at EBR-II is described and example results are presented. It is concluded that the universal network provides different feature recognition capabilities than a neural network and that the universal network can provide extremely fast, accurate, and fault-tolerant estimation, validation, and replacement of signals in a real system.

## INTRODUCTION

Networks are computational schemes which can learn the mapping between the inputs and outputs of complicated functions or systems. The architecture of current network applications is loosely inspired by the human brain and because of this fact they are usually referred to as neural networks. It is also often stated that neural networks provide a distributed, non-algorithmic way of learning to map inputs to outputs as opposed to the usual local, algorithmic, non-learning methods. While this may be true in some cases, it is misleading in general because neural networks use algorithms and because many normal techniques are in fact distributed and include steps which are equivalent to learning. This is true of many of the classical linear techniques of optimization such as least-squares.

---

Work supported in part by the U.S. Department of Energy, Office of Nuclear Energy under Contract No. W-31-109-ENG-38.

One of the key features of neural networks is that they can learn from a large number of training examples, something that is not possible in general with classical linear techniques. In this paper we describe the network of a distributed, non-linear, algorithmic technique for learning in general from any number of training examples. Although this technique can also be described by a modification of classical mathematical expressions, it is described as a network in order to provide insights into the meaning of nodes, connections, weights, and output functions both for neural and universal networks. It is hoped that these understandings may lead to using both techniques simultaneously to solve problems that might lie beyond the separate abilities of each.

## NETWORKS

The modern history of networks began in the 1950s. In notable work during this period Rosenblatt investigated two-layer networks for the ability to learn to recognize patterns (1). Two-layer networks were further developed in the 1960s as adaptive signal processors, most notably by Widrow and Hoff (2), but ran into trouble for more extensive uses when they were shown to have some fundamental limitations in terms of logic that they cannot perform as pointed out by Minsky and Papert (3). Single-layer networks store patterns and can act as content-addressable memories. This was shown by Hopfield who analyzed the dynamics of a fully-connected single-layer network by analogy to a physical system (4). Multi-layer networks were shown to be practical by using the back-propagation algorithm for training the network. Among several workers who showed this were Rumelhart, Hinton, and Williams (5). The back-propagation algorithm has been improved for multi-layer networks and shown to be useful for signal validation, for example in the work of Upadhyaya, Eryurek, and Mathai (6). The latest uses of multi-layer networks involve their ability to adapt to changing control conditions, especially those in non-linear systems, as shown comprehensively in an analysis by Narendra and Parthasarathy (7).

Network characteristics are defined by layers, connections, weights, and output functions. Their characteristics can be summarized by the following statements:

- Layers refer to nodes.
- Connections between nodes transmit the output of one node to the input of another node.
- Weights are associated with connections between nodes.
- Output functions reside in each node and combine the inputs and weights from other nodes to produce the nodal output.

- The output functions resident in each node can be designed to recognize particular features present in the inputs to the node.

- After definition of the layers, connectivities, and output functions, the weights are determined by training the network to match inputs and outputs for a large number of reference examples.

Understanding the output functions at each node is the key to understanding and controlling neural network performance. Each node recognizes some feature or pattern of the inputs presented to it as weighted by their respective connections. The weighted inputs are often simple products and the output functions are often threshold activation functions although specific combinations of weights and inputs and a complicated output function may be necessary to recognize specific features or patterns. The design of the output function of each node may therefore be a very application-specific and network-architecture-specific exercise. Once the architecture, weighting scheme, and output functions are designed then the weights of each connection are determined in a training session where a large number of reference examples are successively presented to the input layer. A computerized search selects the set of weights that simultaneously matches the inputs and outputs of all the reference samples within specified convergence criteria. Initial values of the weights must be supplied for the search to begin. The reference examples used to train a neural network and determine the connection weights are not explicitly used in application of the network.

The continual development of multi-layer neural networks has led to their use in a variety of applications even though these networks require explicit separation of input and output variables, definition of architecture and output functions, and initial values for connecting weights. These are all application-specific characteristics. Furthermore, these networks are not known for their quantitative accuracy.

The universal network is a particular kind of network that is extremely accurate in a quantitative sense and avoids all of the difficulties with designing and determining appropriate input variables, output variables, layers, connectivities, weights, output functions, and convergence criteria that are attendant with neural network applications. It has a partially designated architecture with two hidden layers of known functionality which is completed in its entirety by the reference examples. It does not require separation of inputs from outputs although they can be separated. The reference example values are used explicitly for many of the connection weights, and to determine the

connection weights not explicitly defined. There is no need for initial weight values or iterative calculations.

## A SIMPLE LINEAR MODEL

Figure 1 shows two good data points and one bad data point for a simple linear system which is expected to follow Y = AX + B. The the first two points are good and result from A = 1 and B = 0. The third data point is the faulty one. Specifically these data points are:

X1 = 1 and Y1 = 1
X2 = 2 and Y2 = 2
X3 = 3 and Y3 = 2

These data points can be modeled using a least-squares technique, we find that A = .500 and B = .667 for the data of Figure 1. These values of A and B are far from the correct values and unless we have prior knowledge about the acceptable ranges for A and B or about the possible errors in their measurements we have no way of determining whether they are acceptable.

Figure 1 also shows the results of applying a universal network to the data and finding A = .955 and B = -.150. These values of A and B are much closer to the correct values and illustrate the fault-tolerance of the universal network. The remainder of this section describes how these values for A and B were obtained.

The first step in applying the universal network to this system uses four examples of Y = AX + B that encompass the expected range of X, Y, A, and B values and generally characterize the system.

Example 1 is Y = 5X/3 and the specific values X, Y, A, and B are:

Y = Y11 = 5/3 when X = 1
Y = Y21 = 10/3 when X = 2
Y = Y31 = 15/3 when X = 3
A = A1 = 5/3 independent of X
B = B1 = 0 independent of X

Example 2 is Y = X + 1 and the specific values X, Y, A, and B are:

Y = Y11 = 2 when X = 1
Y = Y21 = 3 when X = 2
Y = Y31 = 4 when X = 3
A = A1 = 1 independent of X
B = B1 = 1 independent of X

Example 3 is Y = X - 1 and the specific values X, Y, A, and B are:

$$
\begin{aligned}
Y &= Y11 = 0 && \text{when } X = 1 \\
Y &= Y21 = 1 && \text{when } X = 2 \\
Y &= Y31 = 2 && \text{when } X = 3 \\
A &= A1 = 1 && \text{independent of } X \\
B &= B1 = -1 && \text{independent of } X
\end{aligned}
$$

Example 4 is Y = X/3 and the specific values X, Y, A, and B are:

$$
\begin{aligned}
Y &= Y11 = 1/3 && \text{when } X = 1 \\
Y &= Y21 = 2/3 && \text{when } X = 2 \\
Y &= Y31 = 3/3 && \text{when } X = 3 \\
A &= A1 = 1/3 && \text{independent of } X \\
B &= B1 = 0 && \text{independent of } X
\end{aligned}
$$

Three explicit values of Y are chosen as input variables, and two explicit values of A and B are chosen as output values from each of the four examples for the reference library of the universal network. The choice of three input variables determines that the number of nodes in the first layer of the universal network is three. The choice of four examples in the reference library determines that the number of nodes in each of the two hidden layers is four. The total number of five input plus output variables determines that the number of nodes in the output layer is five. The output layer of the universal network provides estimates of all input variables plus constructions of all output variables.

The universal network for the above situation is shown in Figure 2. In this figure the conventions are used that ellipses enclose the changing output values of each node in the network and that rectangles contain the fixed weights for each connection. It can be easily seen that the reference examples provide all the weights between the input layer and the first hidden layer, and all the weights between the second hidden layer and the output layer.

The input layer of the universal network does nothing but display the input values, Y1 through Y3, and pass them on as output values.

The first hidden layer of the universal network determines features of the input layer that correspond to features in each column of input variables in the reference library. The particular columns of input values from the reference library are the weights, Y11 through Y34, displayed in the rectangles just to the left of each node. The output function in each node is a parametric function with continuous output values, S1 through S4, between zero and unity. The parameters of these functions are completely determined by the range of values in each row of the

reference library.

The second hidden layer of the universal network determines that combination of features present in each column of input variables in the reference library which are necessary to model the input layer. The output function in each node of this layer produces a simple sum of products, C1 through C4, of the inputs and weights received from the previous layer. The weights, W11 through W44, in the rectangles just to the left of the second hidden layer nodes are determined by training the network to produce a value of unity for C1 and values of zero for C2, C3, and C4 when the first set of input values from the reference library is presented to the input layer, to produce a value of unity for C2 and values of zero for C1, C3, and C4 when the second set of input values from the reference library is presented to the input layer, and so on as each set of values from the reference library is presented to the input layer. This is not an iterative process but instead uses the solutions of an exactly determined set of simultaneous equations and is therefore a very fast calculation.

The output layer of the universal network provides estimates, Y1′, Y2′, and Y3′, of the input layer variables plus constructions, A and B, of all the output variables. The output function for this layer is a simple normalized sum of products of inputs and corresponding weights. In Figure 2 the weights, Y11 through Y34, A1 through A4, and B1 through B4, in the rectangles just to the left of the nodes in the output layer are all derived from the rows of input and output values in the reference library.

Figure 3 shows the numerical results of applying the universal network values above to the data of Figure 1. The input variables are estimated to have values somewhat different than those presented to the input layer, indicating that there is some uncertainty present. The output variables are found to have values of A = .955 and B = -.150 which lie close to the correct values. It can be easily surmised here that the universal network is dominated by the good data points and produces a somewhat fault-tolerant estimate of the model parameters A and B.

## UNIVERSAL PROCESS MODELING AT EBR-II

In the last section of this paper the application of a universal network to variables and parameters with specific meanings in terms of a model of a linear system was outlined. However, the universal network can be applied in a similar manner to any lists of numbers. The universal network can therefore be extended to applications involving large numbers of inputs, outputs, and reference library examples from any source. In doing so it is

usually found that the accuracy and fault-tolerance of the output layer values are increased dramatically over the simple example analyzed above. The main work of a large application is in providing an efficient man-machine interface using menus and graphics that facilitate the identification, manipulation, and control of input variables, output variables, and reference library examples. These tasks are especially cumbersome when large numbers of variables are involved, especially in a real-time environment.

A real-time environment also imposes the necessity to estimate uncertainties in each of the output layer values at each instant of time. This uncertainty analysis is different than a time-series analysis of a sequence of output layer values compared to a similar sequence of actually measured values. Fortunately, an uncertainty analysis of the required type can be performed by the universal network because of the fact that the input layer values are estimated by the output layer. Furthermore, methods of determining correlations between pairs and higher multiplicities of variables has also been developed. The universal network when embedded in software providing the necessary man-machine interface and tools for analysis just described is referred to as universal process modeling software.

Universal process modeling software has been installed on a dedicated UNIX workstation environment at EBR-II. This installation typically receives approximately 40 signal values each second from the reactor system and from the primary, secondary, and steam heat transport systems. A typical list of signals is shown in Table 1. Usually a reference library of approximately 200 examples is used to estimate and validate all signal values each second.

Results obtained from this installation are used in Figures 4 through 7 to illustrate the accuracy and fault-tolerance of this application for measurements of power and pressure. The reference library for these illustrations was obtained from a reactor startup beginning 15 November 1990 and sampled every 10 minutes for 40 hours. After a reactor shutdown another startup occurred beginning 25 December 1990 and this second startup is modeled by the first startup, also every 10 minutes for a period of 40 hours. This modeling application is only for illustrative purposes because normally the reference library would be obtained from a number of examples of startups and full power operations. While Figures 4 through 7 display trends over a forty hour period it is well to keep in mind that the universal process modeling software created 144,000 separate and distinct models of the simultaneous values of 43 signals during this time period.

Figure 4 shows the actual and modeled reactor power for the 240

examples modeled. The modeled signal is virtually indistinguishable from the actual signal on the scale used in this figure. The vertical line near the center of the graph indicates the time when full power was first achieved.

Figure 5 shows an expanded scale of the actual and modeled reactor power after startup for the last 120 examples modeled along with the required limits for validity. The reactor power is valid for this entire time period.

Figure 6 shows the actual and modeled upper plenum pressure for the 240 examples modeled. This signal is quite different from the reactor power previously shown in Figure 4 and yet is modeled simultaneously with the reactor power. The vertical line near the center of the graph indicates the time when full power was first achieved. The modeled signal of Figure 6 can be easily seen to diverge from the actual signal after full power is reached and, as later illustrated in Figure 7, can be determined to be invalid until approximately the last 5 hours of the time period modeled. Invalid in this case may mean that the primary flow was not set correctly or that the pressure sensor is reading incorrectly. Whatever the cause, the problem was corrected by the end of the modeling period illustrated.

Figure 7 shows an expanded scale of the actual and modeled upper plenum pressure after startup for the last 120 examples modeled along with the required accuracies for validity. The invalidity of the upper plenum pressure and its correction can be seen clearly in this graph by comparing the actual signal to the signal validation limits.

## SUMMARY AND CONCLUSIONS

Conventional neural networks provide a content-specific mapping of inputs to outputs for complicated functions, processes, or systems. The content-specific aspects of neural networks are the number of layers of nodes, the number of nodes, the connections between nodes, and the output function present in each node. Weights for each connection are derived by training the network to provide known mappings of inputs to outputs for a large number of examples. Each of the nodes provides recognition of a specific feature contained in the inputs presented to the node as modified by the corresponding weights of each input connection. Feature recognition is therefore dependent on both the architecture and output functions used in the network design. Different processes and systems usually require widely different neural networks.

Universal networks provide a generic mapping of actual inputs to

a combination of estimated inputs plus constructed outputs for any function, process, or system. The separation between inputs and outputs is arbitrary. The number of layers of nodes is always four. All adjacent layers are fully connected. The number of nodes is fixed by the number of inputs plus outputs and by the number of examples used to train the network. The output functions in the first, third, and fourth layers are completely defined. The output function of the second layer is parametrically defined and is determined by the set of examples used to train the network. The weights between the first and second layers and between the third and fourth layers are taken directly from the set of examples used to train the network. The weights between the second and third layers are determined by training the network to perfectly identify and estimate the set of training examples. The second layer provides recognition of features of the input layer that are contained in each of the training set examples. The third layer provides the combination of features among all the training set examples that are necessary for the output layer to estimate the input layer as closely as possible. The output layer provides an estimate of the input layer plus a construction of all output variables. A complete uncertainty analysis of all output layer values can be dynamically provided for each instance of an input layer. Correlations between pairs and higher multiplicities of input and output variables can be calculated.

The feature recognition aspects of a neural network compared to a universal network comprise some of the key differences between them. A neural network requires explicit design of feature recognition while a universal network uses whatever features are present in the set of examples used to train the network. The features that are present in a set of examples are usually a combination of fundamental features. Using a neural network to identify the fundamental features while simultaneously using a universal network to identify combinations of fundamental features may offer a powerful overall technique for analyzing complicated processes.

The universal network is applicable to any process or system for which examples of operation exist or can be simulated. Its application is illustrated for a simple linear system with 5 variables and for a real system at EBR-II with 43 variables. It is concluded that the universal network can be embodied in universal process modeling software to provide uncertainty analyses and correlation analyses, and extremely fast, accurate, and fault-tolerant estimation, validation, and replacement of signals in a real system.

Table 1. EBR-II Signals

| | | |
|---|---|---|
| 1 | Reactor Power Nuclear | MW |
| 2 | Heat Balance Primary System | MW |
| 3 | Heat Balance Secondary System | MW |
| 4 | Heat Balance Steam System | MW |
| 5 | Wide Range A Linear Power Level | % |
| 6 | Wide Range B Linear Power Level | % |
| 7 | Wide Range C Linear Power Level | % |
| 8 | Upper Plenum Probe Pressure 521A | psi |
| 9 | Upper Plenum Probe Pressure 521B | psi |
| 10 | Primary Pump 2 Out Flow 521B Voltage | MV |
| 11 | Low Pressure Plenum 2 Flow 513B Voltage | MV |
| 12 | Low Pressure Plenum 2 Flow 541B Voltage | MV |
| 13 | Primary Total Out Flow 541E Voltage | MV |
| 14 | Reactor Outlet Flow Rate 541E | gpm |
| 15 | Low Pressure Plenum Sodium Temp. 540AR | Deg. F |
| 16 | Low Pressure Plenum Sodium Temp. 540AS | Deg. F |
| 17 | Low Pressure Plenum Sodium Temp. 540AV | Deg. F |
| 18 | High Pressure Plenum Sodium Temp. 540AT | Deg. F |
| 19 | Reactor Outlet Temp. 1534CF | Deg. F |
| 20 | Reactor Outlet Temp. 503 Bailey | Deg. F |
| 21 | Reactor DT 506 Bailey | Deg. F |
| 22 | Reactor DT Based On 1534CF Outlet | Deg. F |
| 23 | Reactor DT Based On Average S/A Outlet | Deg. F |
| 24 | Reactor DT Test Calculation | Deg. F |
| 25 | IHX Secondary Outlet Temp. 533AA | Deg. F |
| 26 | IHX Secondary Outlet Temp. 546A | Deg. F |
| 27 | Superheater Sodium Inlet Header 508A | Deg. F |
| 28 | Superheater Sodium Inlet Header 546H | Deg. F |
| 29 | Superheater Sodium Inlet Header 546J | Deg. F |
| 30 | Evaporator Outlet Header North 508D | Deg. F |
| 31 | Evaporator Outlet Header North 546AM | Deg. F |
| 32 | Evaporator Outlet Header South 508C | Deg. F |
| 33 | Evaporator Outlet Header South 546AL | Deg. F |
| 34 | Generator MW 642 | MW |
| 35 | Average Bulk Sodium Temp. 501AA-X | Deg. F |
| 36 | Average Reactor Inlet Temp. 540AR-S-V | Deg. F |
| 37 | Average Subassembly Outlet Temp. | Deg. F |
| 38 | Average Upper Plenum Probe Temp. | Deg. F |
| 39 | Average IHX Primary Outlet Temp. | Deg. F |
| 40 | Average Superheater Sodium Inlet Temp. | Deg. F |
| 41 | Average Evaporator Sodium Outlet Temp. | Deg. F |
| 42 | Average Evaporator Feedwater Inlet Temp. | Deg. F |
| 43 | Average Evaporator Steam Outlet Temp. | Deg. F |

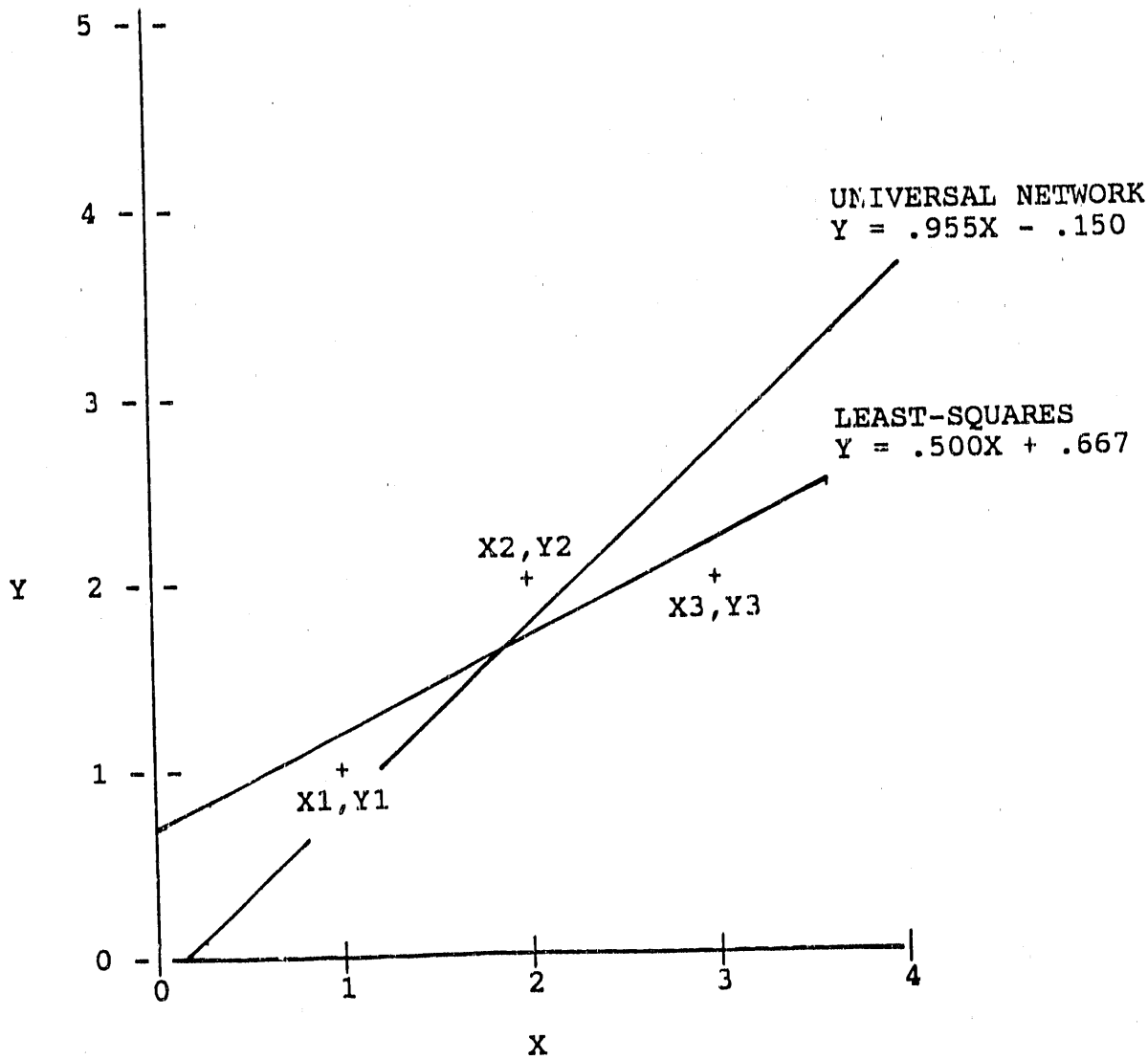


Figure 1. Universal network and least-squares estimates of two good data points and one bad data point for a system modeled by Y = AX + B.

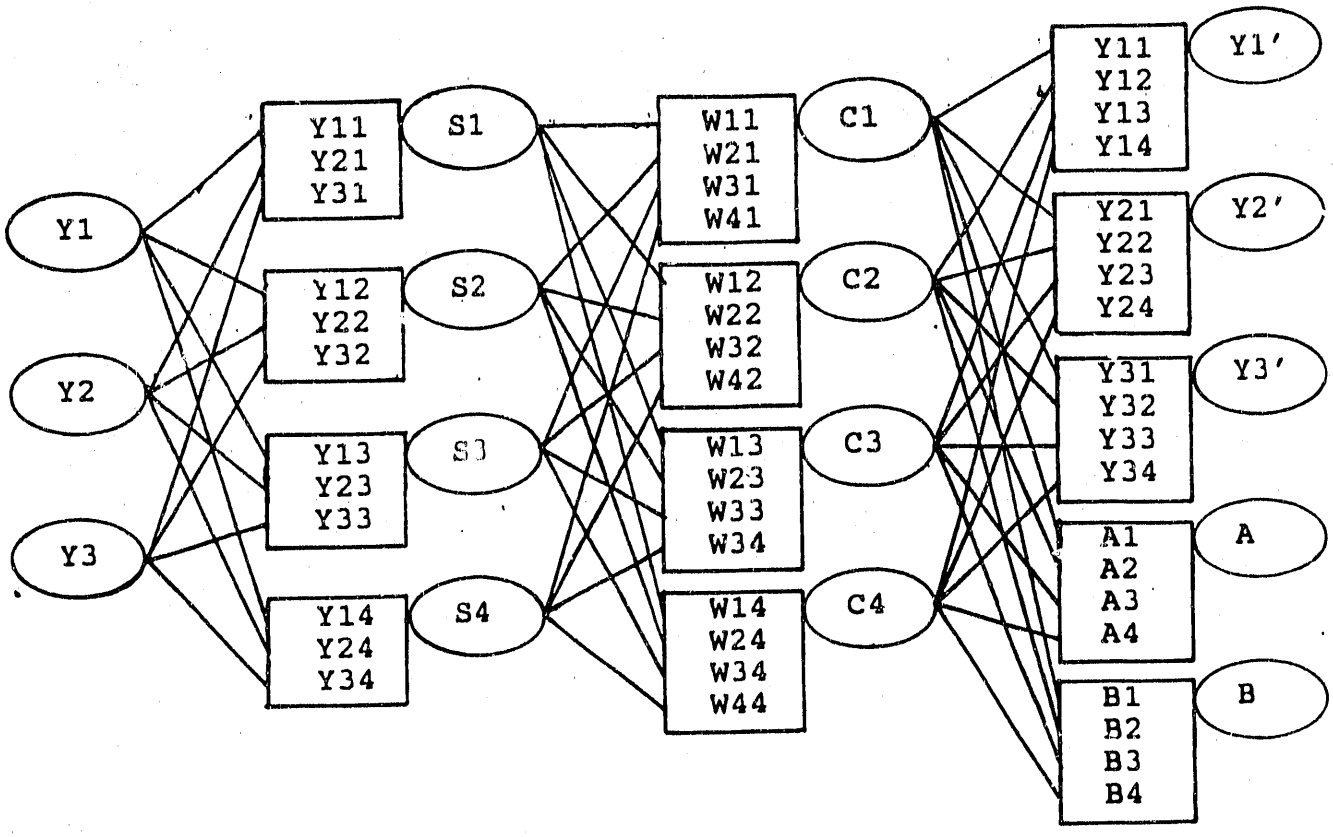


Figure 2. A universal network with three inputs, four sets of examples, and two outputs for the solution of the problem illustrated in Figure 1. See text for definitions of symbols.

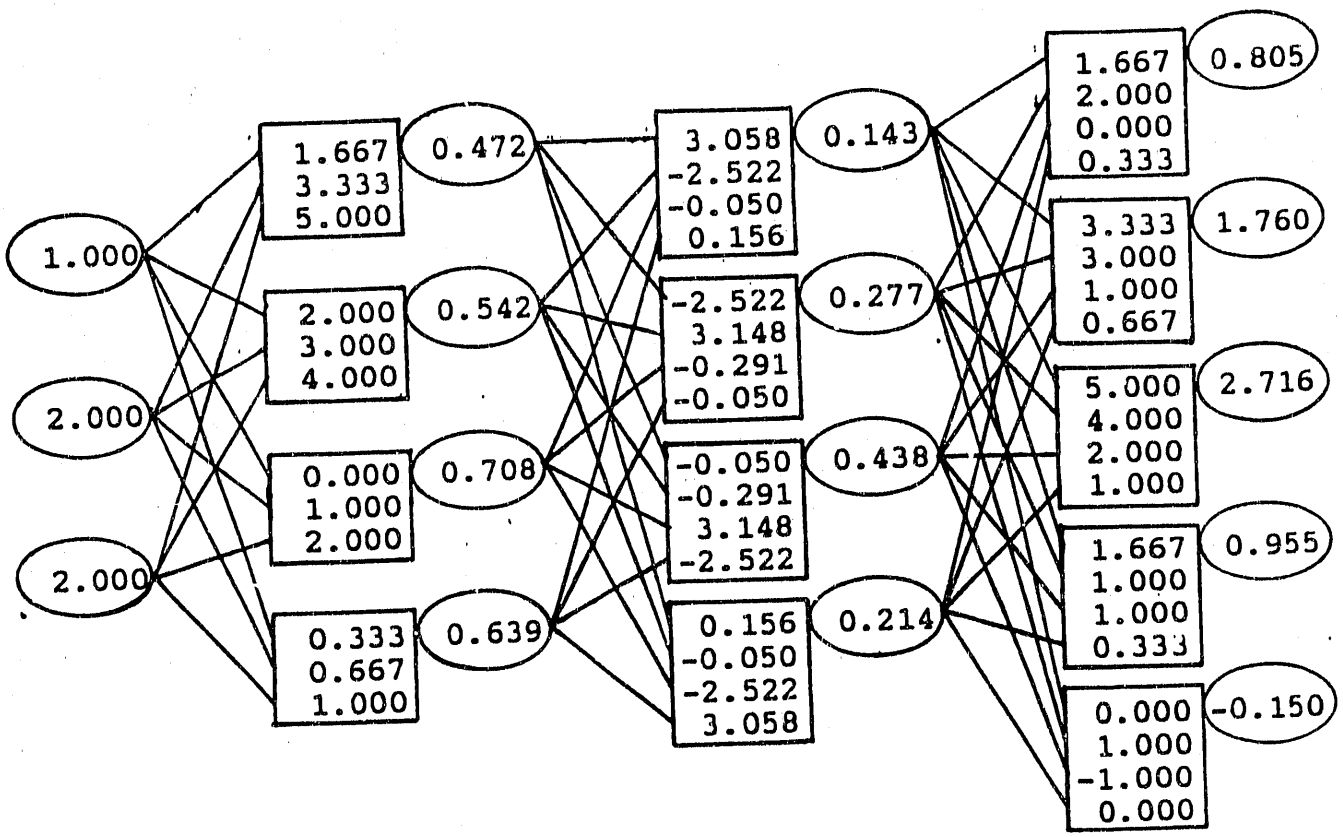


Figure 3. Nodal values and weights for the universal network of Figure 2 applied to the solution of data in Figure 1. See text for source of numerical values.

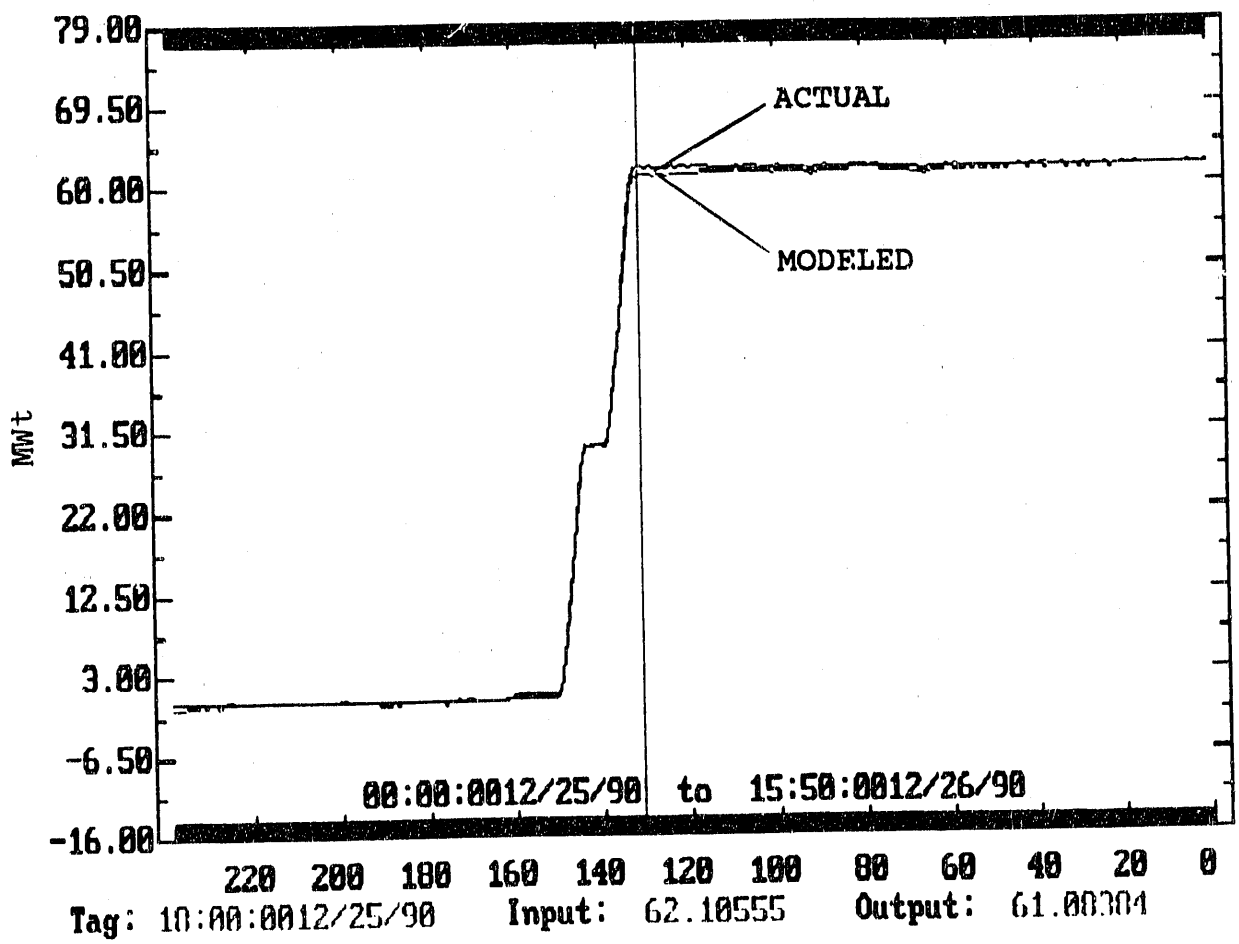


Figure 4. The actual and modeled reactor power every 10 minutes for 40 hours of EBR-II operation on 25 and 26 December 1990. The reference library was obtained from similar data on 15 and 16 November 1990.

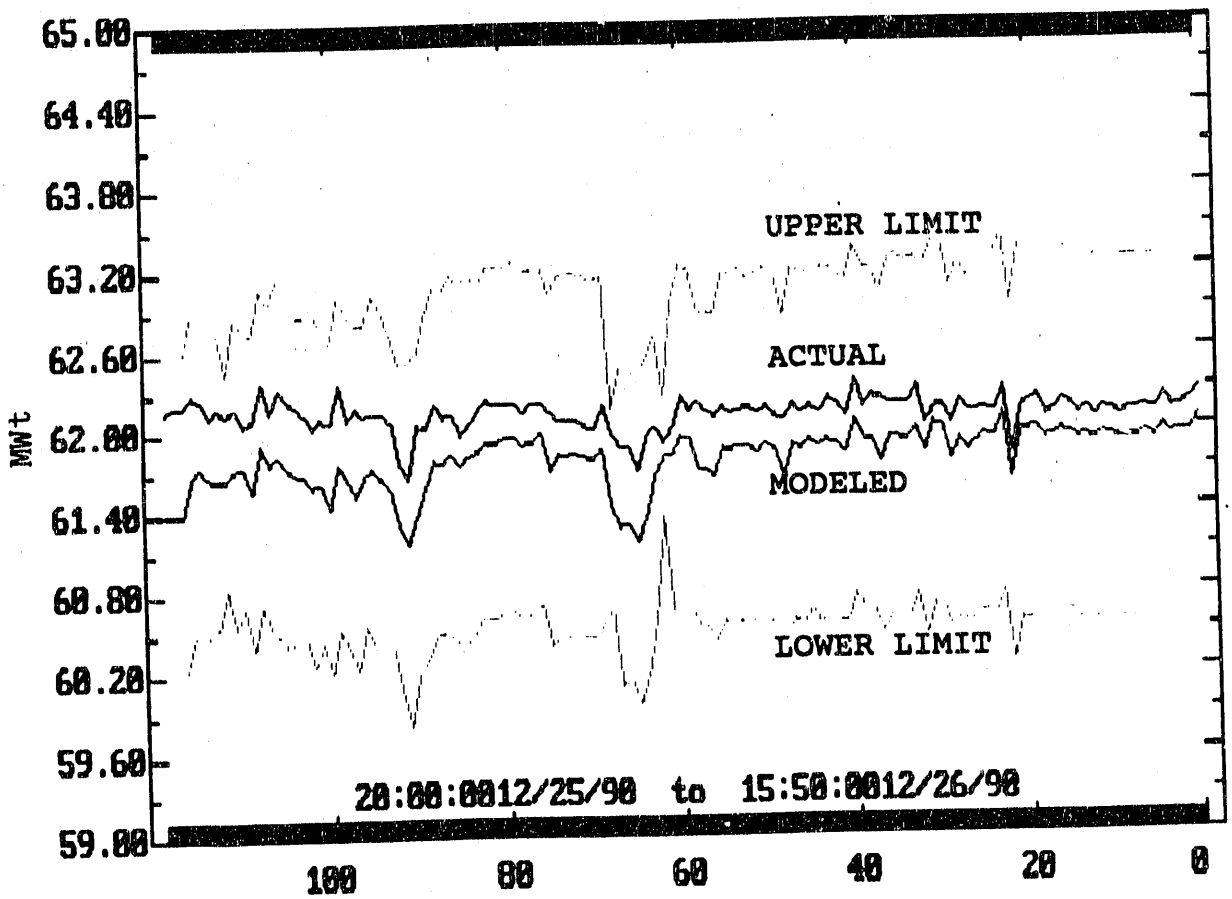


Figure 5. An expanded view of the right half of Figure 4 with upper and lower validation limits.

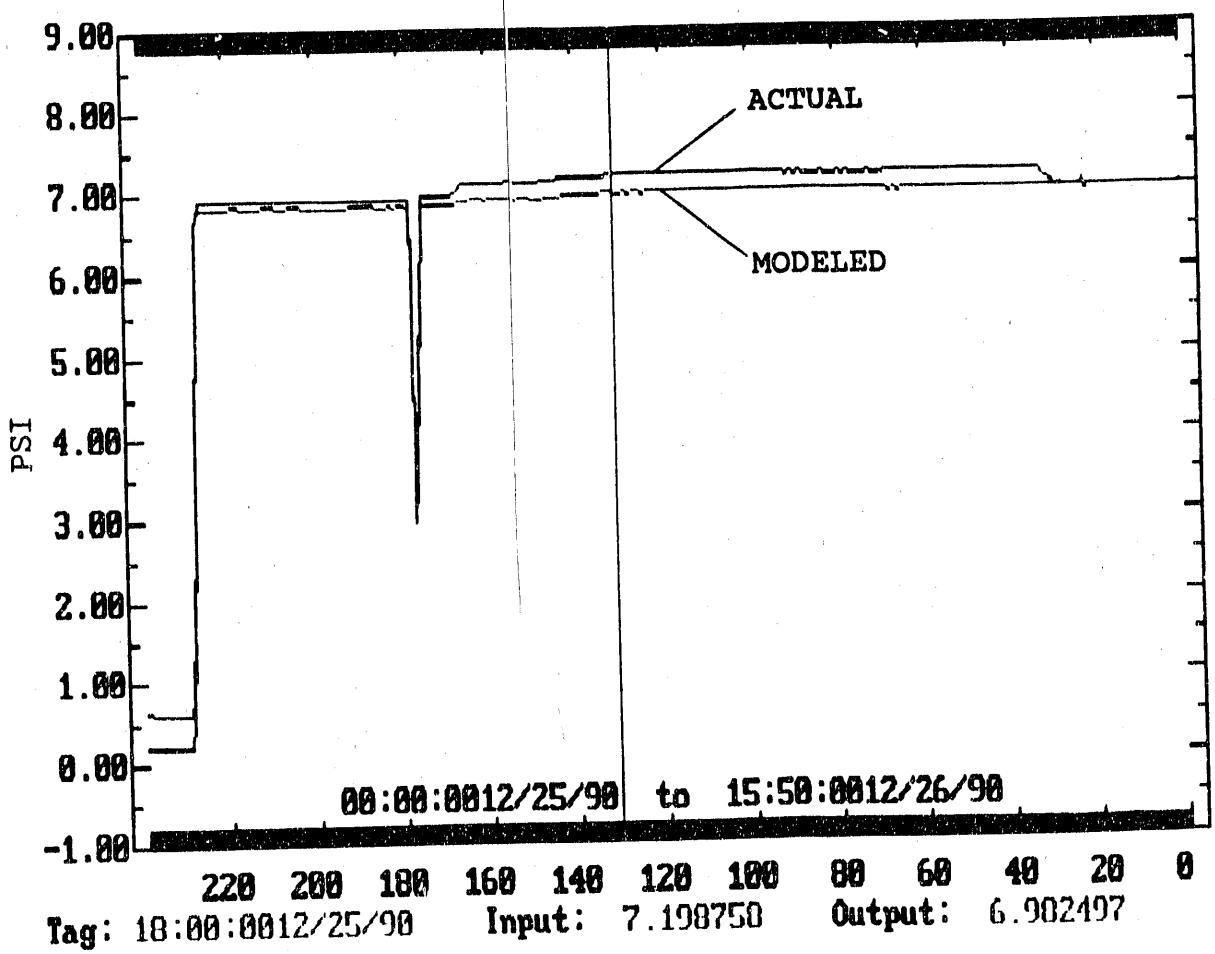


Figure 6. The actual and modeled upper plenum pressure every 10 minutes for 40 hours of EBR-II operation on 25 and 26 December 1990. The reference library was obtained from similar data on 15 and 16 November 1990.

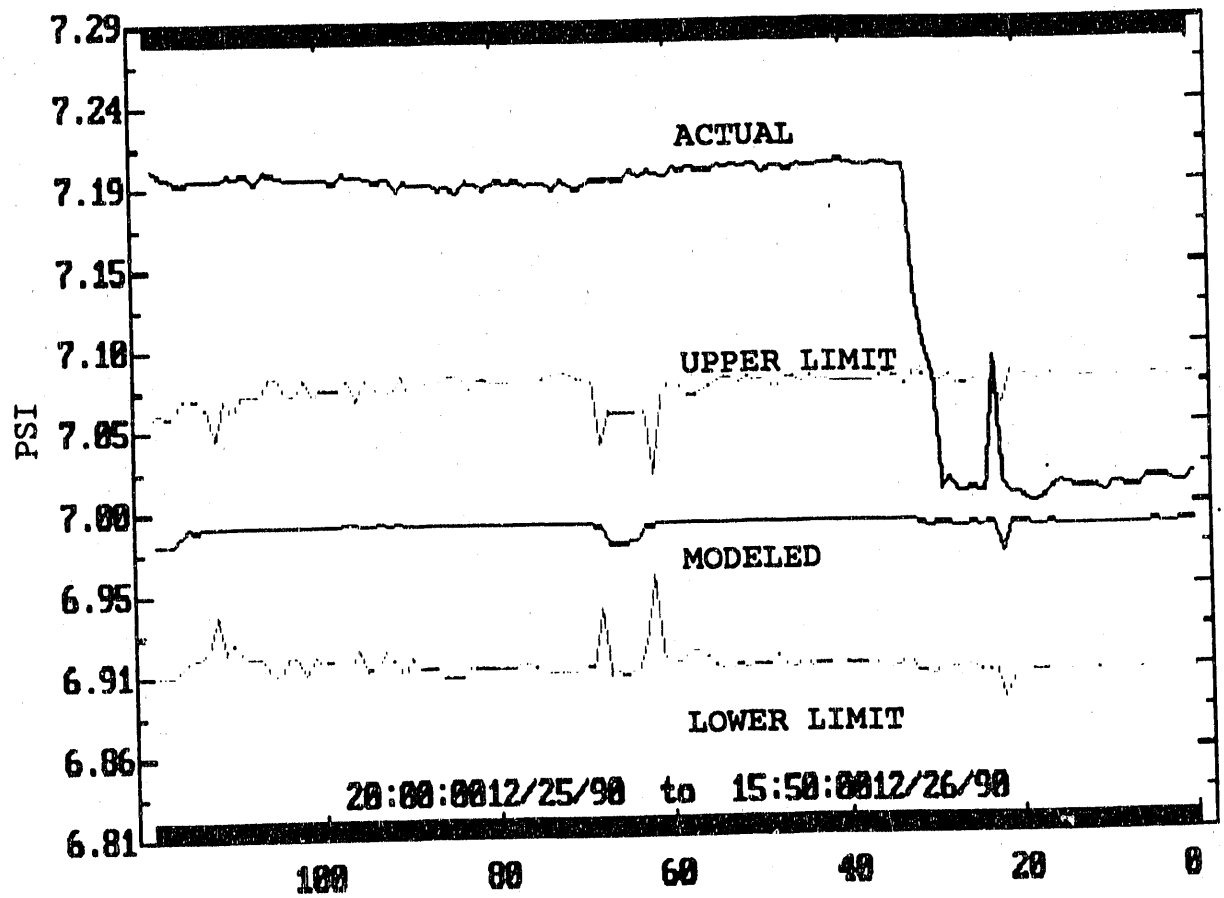


Figure 7. An expanded view of the right half of Figure 6 with upper and lower validation limits.

## REFERENCES


1. F. Rosenblatt, "The Perceptron: A Probabalistic Model for Information Storage and Organization in the Brain," Psychological Review, Vol. 65, 1958, pp. 368-408.

2. G. Widrow and and M. E. Hoff, "Adaptive Switching Circuits," IRE WESCON Convention Record, 1960, pp. 96-104.

3. M. A. Minsky and S. Papert, Perceptrons, MIT Press, 1969.

4. J. J. Hopfield,"Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proceedings of the National Academy of Sciences, Vol. 79, 1982, pp. 2554-2558.

5. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in D. E. Rumelhart and J. L. McClelland, Parallel Distributed Processing, MIT Press, 1986.

6. B. R. Upadhyaya, E. Eryurek and G. Mathai, "Application of Neural Computing Paradigms for Signal Validation," Proceedings of the 7th Power Plant Dynamics, Control & Testing Symposium, Knoxville, Tennessee May 15-17, 1989.

7. K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," IEEE Transactions on Neural Networks, Vol. 1, 1990, pp. 4-27.

# END

DATE
FILMED
6/05/92