N93-11933

Design Development of a Neural Network-based Telemetry Monitor

Michael F. Lembeck, Ph.D.
Space Industries, Inc.
711 W. Bay Area Blvd., Suite 320
Webster, TX 77598
(713) 338-2676

Abstract

This paper identifies the requirements and describes an architectural framework for an artificial neural network-based system that is capable of fulfilling monitoring and control requirements of future aerospace missions. Incorporated into this framework are a newly developed training algorithm and the concept of cooperative network architectures. The feasibility of such an approach is demonstrated for its ability to identify faults in low frequency waveforms.

1.0 Introduction

As aerospace systems become more complex, the need to quickly predict, identify, and correct faults becomes more critical to mission success. Future systems on board spacecraft will have requirements for longer lifetimes, higher reliabilities, and lower maintenance than previously encountered. To meet these requirements, Artificial Intelligence (AI) tools can assist human operators in anticipating failures in equipment from trends in sensed data. This paper examines the utility of artificial neural networks and the architectures and implementations required for monitoring real-time sensor telemetry signals.

Several methods are available for training neural networks to perform pattern recognition tasks. The Self-Scaling Conjugate Gradient method presented here is shown to be applicable for training neural networks to solve the telemetry signal monitoring problem. This method also demonstrates exceptional performance, measured in terms of network convergence time, when compared to other available training methods.

To monitor signal waveforms and generate an output indicating waveform "health" four "cooperative" neural networks are used. These networks, each monitoring a specific "part" of the signal of interest, are shown to be capable of detecting faults in low frequency waveforms.

1.1 Problem definition

A high priority in the design of real-time monitoring systems is the reduction of large amounts of sensor telemetry data into an immediately useable form for human operators and machine interpretation. Serial instruction-based computation devices monitoring simple limits on telemetry data will be replaced with machines capable of parallel operations which are more adept at pattern identification. Artificial neural networks, which have been shown to be quite adept at identifying patterns in data will play a large role in this effort.[13,14,16]

Expert systems have been developed to perform "intelligent" control or management of system configurations (i.e. the optimum scheduling, switching, or control of devices, given a set of resource constraints and available modes).[4,7] Sensor data indicating the current state of the system is tested against knowledge gathered from experts familiar with the given system's operation. This knowledge may take the form of rules specifying how the system should be configured for nominal operations as well as in the event of a system fault.

Unfortunately, conventional expert systems, running on today's hardware, cannot respond in real-time (defined here as on the order of microseconds to milliseconds) to system faults which require detailed analysis and immediate reconfiguration.[2,20] For such applications, a network can be designed to recognize specific sensor data patterns, associate them with a specific output (much like the assertion of a "consequent" of a conventional rule in an expert system) and

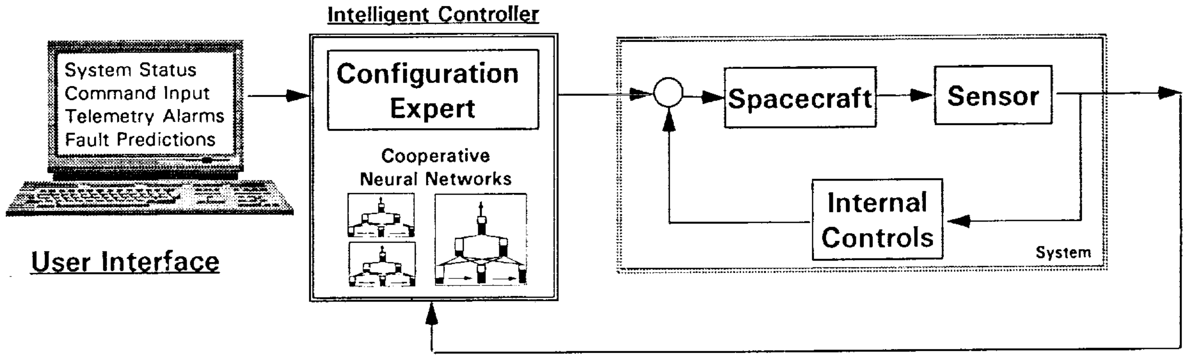


Figure 1. Neural Network-based telemetry monitor reference system.

output the result to some configuration manager or controller interface. Implemented in hardware, neural network response times may approach those required for real-time control.[10,19]

The reference system illustrated in Figure 1 serves as a model for developing and evaluating applications of artificial neural networks to monitoring and control problems. An external configuration manager commands the system to achieve the desired result. Sensors determine the actions taken by the system in response to the controlling input. Additional internal controls may be employed to achieve localized regulation of the system. Sensor data is also reported to an external telemetry monitor (perhaps viewed by a human operator) and routed back to the configuration manager.

2.0 Artificial Neural Networks

An artificial neural network can be defined as a "parallel interconnected network of simple (usually adaptive) elements and their hierarchical organizations which are intended to interact with the objects of the real world in the same way that biological systems do."[9] The simplest model of an artificial neuron is drawn in Figure 2.

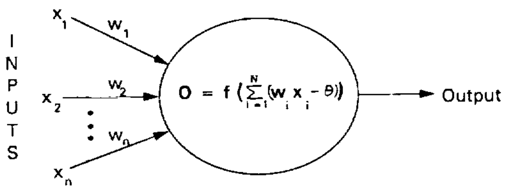


Figure 2. Simplified artificial neuron.

The illustrated simplified artificial neuron receives an arbitrary number of input signals, x_i, each weighted or modulated by a gain, w_i. These weighted signals are then summed and characterized by an interior threshold or bias, θ. Finally, the result is passed through a mapping function, f, to generate an output signal, O. The mapping function generally represents some continuous nonlinear (hard limiter, threshold logic, or sigmoid) "activation" function through which the weighted sum is passed before being expressed as an output.

Collections of connected neurons, making up a single layer neural network, can be constructed by connecting several inputs to more than one neuron. It is customary to index connection weights as w_{ij}, meaning the weight value on the connection from node *i* in a lower layer to node *j* in an upper layer.

In the single layer case, the output functions, O_k, may be computed as:

$$O_k = f_k(\sum w_{ij} x_i - \theta_k) \qquad 1$$

Network architectures composed of several layers of nodes are possible. Hornik has shown that these multilayer, feedforward networks can be used as universal approximators for various functions.[8] Cybenko has shown that arbitrary decision regions (stored patterns) can be arbitrarily well approximated by a continuous feedforward neural network with only a single hidden layer and any continuous sigmoidal $(f(x) = 1/(1 + \exp(-x))$ mapping function.[1] This important result is the justification for concentrating on the so-called "three layer feedforward network."

In this type of network, input signals are passed through an input layer which scales the signals into the range $0.0 \leq x \leq 1.0$ for use by the network. The scaled signals are multiplied by connective weights and input to a middle or "hidden" layer. At each node all weighted values are summed and passed through a sigmoid nonlinearity for output to the next layer. The process is then repeated in the next highest layer of the network in order to produce the desired output.

2.1 Training Neural Networks

Patterns may be stored in a network by varying the connective weights between network nodes until the desired associative outputs are generated in response to the presentation of a training input pattern. Methods by which the weights are iteratively updated are called "training" algorithms. Introductory descriptions of these algorithms and several other neuron models and network architectures can be found in the literature.[9,12]

The training of an artificial neural network to properly generalize and associate a desired output pattern when presented with a given input pattern can be posed as a multivariable optimization problem based on the "generalized delta rule."[17] First, an objective or error function must be defined which represents how well the network has learned its task and is generally a function of the target and actual output values. The optimization objective is to find a set of network weights which minimizes the error function for all pattern presentations.

One way of accomplishing this is to minimize the sum of the squares of the difference between output node values and the desired target values for each output node. In the following expressions, *k* refers to nodes in the output layer, *j* refers to nodes in the hidden layer, *i* refers to nodes in the input layer, and *p* refers to the pattern presentation sequence number. For each input pattern/output target pair, Equation 2 defines a common variation of an error function, E_p, used as a starting point for defining training algorithms for a neural networks. Here t_{pk} is the *p*'th target for the *k*'th node and o_{pk} is the output for that node.

$$E_p = \sum_{k=1}^{K}(t_{pk} - o_{pk})^2 \qquad 2$$

A ***"presentation"*** or ***"function evaluation"*** takes place when a single pattern is presented to a network's input nodes and this input is then fed forward through the network to produce an output at the output nodes. ***"Gradient"*** or ***"derivative evaluations"*** are computed when the error resulting from this presentation is fed-back through the network and weight function gradients are calculated for each weight and bias. After all patterns in a training set have been presented as inputs to a network, a ***"step"*** or ***"epoch"*** is then said to have passed. Depending on the training algorithm, ***"weight updates,"*** or the modification of the connecting weights, may take place after a step or epoch.

2.2 Training Methods

Given the definition of an error function, an expression for incrementally updating the connection weights to minimize this function can be derived. The weight update is formulated as the derivative of the error with respect to each weight and proportional to some negative constant as given by Equation 3.

$$\Delta_p w_{kj} = -\eta \frac{dE_p}{dw_{kj}} \qquad 3$$

Various approaches have been taken to this optimization problem. Some of the most successful, and a new more efficient method, are explained below.

2.3 On-line Back-propagation

On-line back-propagation is the name given to the scheme where a weight update is computed after every individual pattern is presented to the network input nodes and the error (Equation 2) for each pattern is minimized. Thus, one function evaluation, one gradient evaluation, and one weight update occurs with every pattern presentation. While not a true descent method, this robust algorithm has been quite successful in solving a large variety of problems.[17,3] Equation 4 illustrates a common weight update formula used for the hidden-to-output layer connections.

$$\Delta w_{kj}(n+1) = \eta\,(t_{pk} - o_{pk})\, f'(net_{pk})\, o_{pj} + \alpha\, \Delta w_{ji}\,(n) \qquad 4$$

Here p is the pattern number, t_{pk} is the target value for a given output neuron, o_{pk} is the neuron's actual output, f' is the derivative of the sigmoid function, and net_{pk} is the sum of all inputs to the neuron coming from the hidden layer. The training rate, η, and the momentum coefficient, α, are fixed for the duration of the training session. "Optimal" values of these two constants are usually determined in an empirical manner for each problem.

Aside from its successes, this method has several drawbacks, especially when applied to problems of large scale. Since η is fixed, too large or too small a step may be taken in the descent direction resulting in a violation of the descent condition or an overshooting of the minimum. In the case of back-propagation with momentum, a fixed α may result in a step being taken in a non-descent direction where any η may result in an increase in the system error. Finally, because a weight update is performed after each pattern is presented to the network, the pattern presentation order can have an effect on the rate of convergence.

2.4 Conjugate Gradient Methods

Conjugate gradient methods have a long history of solving large dimensional problems where other methods fail.[5,18] For the neural network training problem it is desired to iteratively minimize the network error function after each set of patterns has been presented to the network. New values for each connective weight w then are defined in terms of the gradient, g_k, and the search direction, d_k at each step:

$$w_{k+1} = w_k + \alpha_k d_k \qquad 5$$

where

$$d_{k+1} = -g_{k+1} + \beta_k d_k \qquad 6$$

$$\beta_k = \frac{(g_{k+1} - g_k)^T g_{k+1}}{(g_{k+1} - g_k)^T d_k} \qquad 7$$

The choice for α_k is made by finding the minimum value of the error function through successive line searches (finding the minimum of a function along search direction, d_k). When *m* is the number of iterations required to locate a minimum to a given tolerance and *p* is the number of patterns in the training set, each line minimization will require *mp* function evaluations. After a suitable minimum has been found, *p* number of gradient evaluations are then required in order to compute the next search direction.

This study was concerned with methods in which storage and computational requirements are minimized for possible onboard spacecraft applications. Algorithms were considered if only the last search direction and gradient need to be retained from step to step and the overhead for computing iterative search directions is low. Higher order methods (BFGS, quasi-Newton, etc.), require storage of additional arrays in order to build up approximations to the Hessian and sometimes incur a significant overhead in search direction computation.[18]

The primary objective here is to find a method which will allow a neural network to reliably converge to a weight set with a minimum amount of function and gradient calculations. These evaluations are the means by which competing algorithms are judged.

2.5 Self-Scaling Conjugate Gradient Method

In his paper on conjugate gradient methods with inexact searches, Shanno reviews several different conjugate-gradient-type methods for application to several classes of problems.[18] One such method, Shanno's Equation 26a, derived from work carried out by Oren and Spedicato, represents a Self-Scaling Conjugate Gradient (SSCG) algorithm which does not require the storage of additional arrays.[15]

With line minimizations performed as before to find the step length, the new search direction for each iteration is given by Equation 8.

$$d_{k+1} = -\frac{p_k^T y_k}{y_k^T y_k} g_{k+1} - \left(2\frac{p_k^T g_{k+1}}{p_k^T y_k} - \frac{y_k^T g_{k+1}}{y_k^T y_k}\right) p_k + \frac{p_k^T g_{k+1}}{y_k^T y_k} y_k \qquad 8$$

where

$$p_k = \alpha_k d_k \quad \text{and} \quad y_k = g_{k+1} - g_k$$

The SSCG algorithm represents a potentially powerful method for carrying out neural network weight-updating training algorithms. This gradient descent method is less susceptible to certain local minimums. Finally, the order of presentation of the input patterns is not critical.

3.0 Temporal Windowing

One approach to formatting telemetry data for input to a neural network is to sample the input signal at discrete time intervals, scale this into a range acceptable by the network, and feed this signal into the input nodes of the network. If the input layer contains more than one node, successive input nodes may be stimulated with time-delayed samples of the input data. Such an approach is called "Temporal Windowing."[6]

Figure 3 shows an example of three temporal windows of a 1.0 Hz sine wave sampled every 100 ms (10 Hz sampling rate).

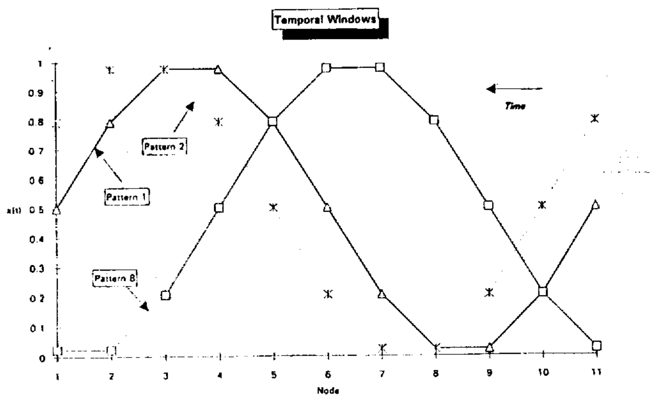


Figure 3. Temporal windows for a 1 Hz sine wave.

Pattern 1 shows the initial snapshot of 11 values making up one complete period of the wave. Pattern 2 represents the next available snapshot, taken 100 ms later. The rightmost data point represents the most recently sampled part of the sine wave. Pattern 8 represents the state of the wave 700 ms after the initial snapshot (Pattern 1). Note that ten patterns are required to show one complete period of the sine wave.

Another way to view temporal windowing is that a sample is taken, displayed to the rightmost input node for a short delay period (100 ms), and then displayed to the node immediately left of the previous node. Figure 4 illustrates this sample, hold, and shift input procedure.

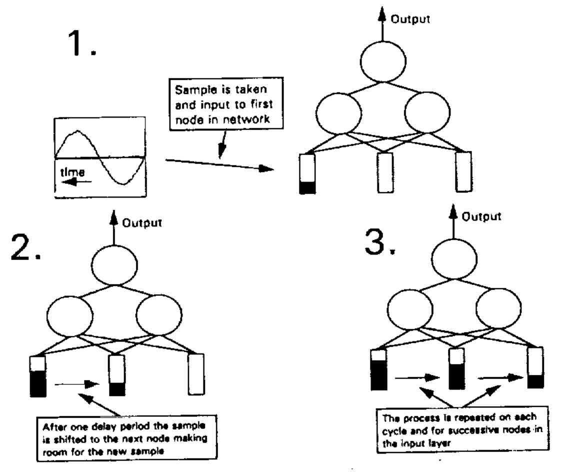

1.
Output
Sample is taken and input to first node in network
time
2.
Output
3.
Output
After one delay period the sample is shifted to the next node making room for the new sample
The process is repeated on each cycle and for successive nodes in the input layer

Figure 4. Sample, hold, and shift sample inputs to input layer nodes.

To monitor the temporal integrity of the input waveforms, other cooperative networks (Figure 5) can be constructed to keep track of the temporal ordering of data being put into and recognized by the signal recognition network. In the example case, the phase angle recognition network can be trained to output an analog value in the range (0,1) corresponding to the phase pattern number of the rightmost node value. Another network is then trained to detect the proper sequence of patterns coming out of the phase recognition network and provide an indicator signal whenever this order varies from the prescribed sequence. A fourth network combines the phase sequence indicator with the waveform recognition indicator to produce an overall indication of signal health.

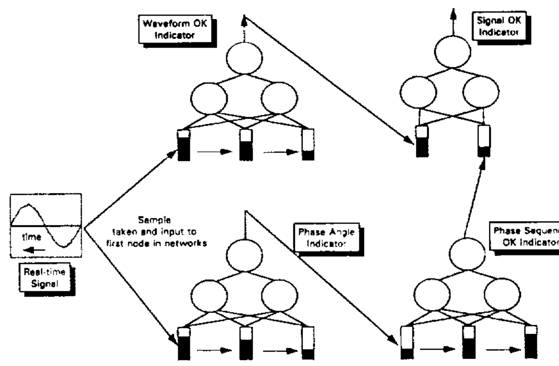


Figure 5. Cooperative signal recognition networks.

4.0 Telemetry Monitor Application

The application presented here is the monitoring of a simple sine wave by a cooperative network set while providing an indication of when it deviates from a nominal amplitude and frequency. This example also demonstrates the utility of the SSCG training method. When an [11,25,1] network was trained with 40 sine wave patterns, a modified backpropagation routine took 21440 function and 21440 gradient evaluations. The SSCG method required 20664 function evaluations and only 2040 gradient evaluations to reach the same level of convergence.

For this test application, it was desired to correctly identify a simple, continuous 1.0 Hz, 0.5 amplitude sine wave and to detect any deviations in amplitude or frequency. Any detected deviations were to be called to attention by the loss of a "good" signal indication.

The simulation takes advantage of the cooperative network concept by using four networks (as illustrated in Figure 5) to determine the input signal's "health." The first net is trained on the temporally windowed input signal waveform and outputs a "good/bad" (generally a one/zero output) signal depending on the waveform's degree of match with the internally represented, previously learned training set. A second net receives the same signal simultaneously and outputs a value corresponding to the phase angle of the input signal. This output value is temporally windowed into a third timing network which looks for a regular, repeating pattern of phase angles corresponding to a good waveform. The fourth and final network has two input nodes. One node receives the good/bad signal from the first signal recognition net and the second receives the good/bad signal from the phase timing network. This control network's single output node then indicates the interpreted state of the original input waveform.

This application made use of a PC-based program to train the four required nets and another program to link the four nets together cooperatively in a user-friendly (Windows 3.0) environment. Reasonable net sizing was determined by a short set of function evaluations studies.

Training sets for each of the four nets were defined as follows. For the signal recognition [11,25,1] net, a set of ten temporally related "bad" sine patterns of amplitude 0.45, ten bad patterns of amplitude 0.55, and 20 good patterns of amplitude 0.5 were constructed. For the phase recognition [11,15,1] net, ten 0.9 Hz bad patterns, ten 1.1 Hz bad patterns, and 20 1.0 Hz good patterns comprised the training set. The [11,10,1] phase timing network training set consisted of five windows of bad random input values, ten windows of bad constant input values in the range from 0.0 to 0.9, and ten windows of properly phased timing values. The forth, [2,3,1] control network was trained with four patterns. Whenever its two input nodes received a good output from the signal and phase timing networks, it would output a good signal (1.0). When the signal network indicated trouble, the voting network was trained to output a bad value of 0.75 indicating failure. Phase timing failures would cause the network to output a bad value of 0.5. The failure of both recognition nets would cause a zero value to be output.

After training to an average system error of less than 1E-4, the weight sets from the four networks were merged together into a cooperative set and loaded into the Windows-based program.

The results of this demonstration are best illustrated by examining the output from the four networks during normal operation and after induced failures (Figures 6 and 7). After an initial startup transient, the signal recognition network outputs a good value of 1.0 for an input amplitude of 0.5. The periodicity of the output value, small though it is, may be the result of incomplete training. If this is so, it can possibly be minimized by using a larger training set and a smaller convergence tolerance. The phase recognition network shows the periodic ramping of the recognized phase angle of the signal, and the phase timing network outputs a good signal for this. The control network's concurring output stays at one for as long as the waveform is correct.

Two cases, where failures are introduced at the five second mark, demonstrate the ability of the network to detect small deviations ($\pm$0.05) in input signal amplitude. Figure 6a shows the signal recognition net's failure signal. Figure 6b shows what happens to the phase angle net's output. In this case, trained only with constant amplitude waveforms, the phase net mistracks the phase angle at the lower amplitude. Figure 6c shows the phase timing net's output, as it loses its timing at the lower amplitude. Note that it might be desired that for all cases of amplitude failures, a phase network would still output a good signal, as it would if it were truly measuring phase. Such a response would be made possible by adding waveforms of varying amplitudes to the training set. In this instance, the control net responded reasonably well (Figure 6d) with the correct output (0.75) for the high amplitude failure, but the low amplitude failure caused the net to oscillate between 0.0 and 0.75 as the phase timing net repeatedly lost lock.

A second test case was performed, this time with frequency failures of $\pm$0.01 Hz injected at the five second mark. The signal recognition network, trained with constant 1.0 Hz frequency patterns indicated failure in an oscillatory manner (Figure 7a). Likewise, the phase network and its cooperative timing network, whose output is plotted in Figures 7b-c, failed to track the failed signal's phase. The resulting oscillating control network's output in Figure 7d diligently tracked the phase timing and signal recognition output as best it could.

5.0 Future Applications

The work presented here demonstrates the feasibility of using artificial neural networks to monitor low frequency real-time processes. Additional studies reported elsewhere have demonstrated the ability to use this network architecture to actually control a single parameter dynamic system (e.g., spacecraft actuator spin rate).[11]

This new application of the SSCG algorithm makes it possible to train and examine several network architectures for a specific application in a relatively short period of time. This is of significant advantage until a more analytical approach to sizing

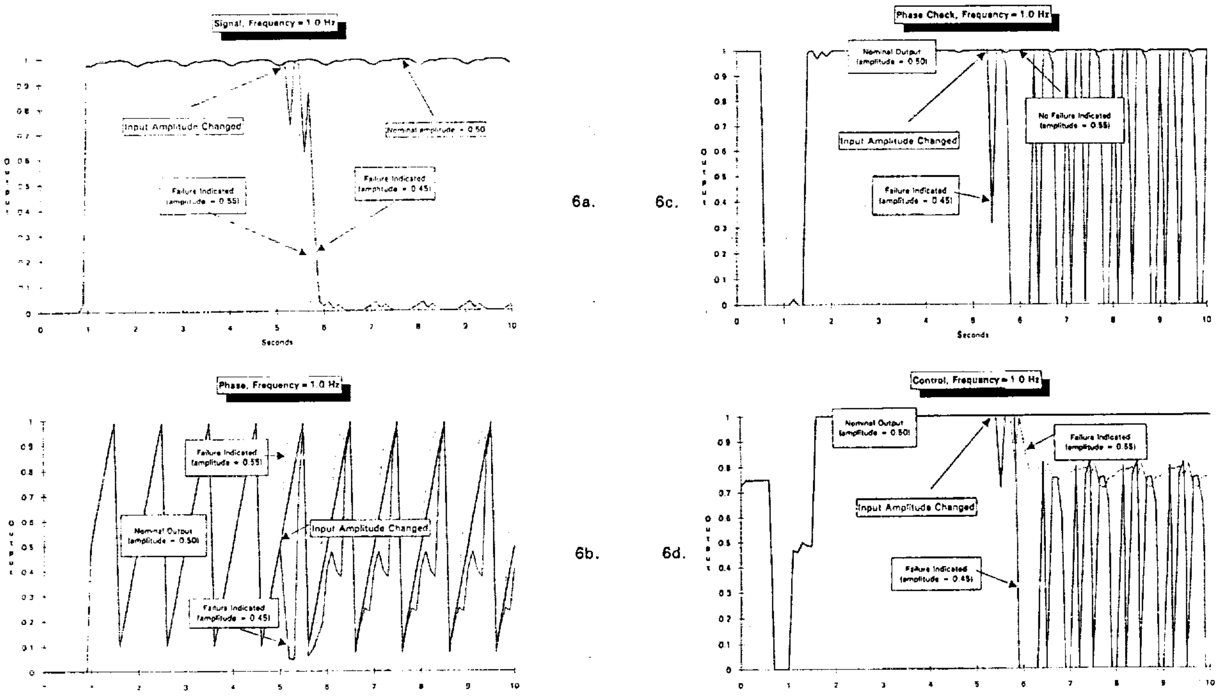


Figure 6. One Hz sine wave with induced amplitude faults.

neural networks becomes available. The availability of neural networks implemented in hardware will also speed up the training cycles and allow larger training sets to be presented in shorter periods of time.

Other topics related to telemetry monitor design are also open to further improvement and investigation. Real-time performance is a critical concern in the development of an intelligent space-borne configuration controller and telemetry monitor for controlling multiple subsystems. With large amounts of time-varying data, the validity and timeliness of conclusions based on instantaneous data is constantly in question. First attempts at using expert systems for real-time applications involved taking a snap-shot of data and using a static expert system to draw conclusions about system health. Conventional pattern-matching paradigms which examine all possible conclusions for the current data values are too slow for most real applications. Yet expert systems may still play a role in real-time controllers.

One approach would integrate a neural network front end with an expert system configuration controller. When performance exceptions are detected by the neural networks, an inference engine might invoke a set of metarules which would focus the attention of the inferencing system on the offending subsystem. The benefit of this approach is that knowledge bases with thousands of rules, properly gathered into smaller, related sets of rules, can be run in real-time.

Before neural network-based systems see everyday operations, the issues of verification and validation will also need to be addressed. When neural networks are called upon to generalize a desired response from an incomplete training set, it must be verified conclusively that the proper generalization was made. Accomplishing this within a finite amount of test time is a difficult issue yet to be fully studied.

Finally, future work in this area should also address the effects of input signal noise on the ability of the signal and phase recognition nets to discriminate their desired waveforms. Noise might also serve as a means to create a deadband or wider acceptance region around some nominal patterns used to train networks. Random amplitude noise, superimposed on top of the repeating, temporally related patterns can make those patterns appear to "cover more space" than the basic set. In this manner, a significantly large deadband effect might be learned by a network with only a small input pattern training set.

Artificial neural networks and their successors will soon find their way into the aerospace engineer's box of tools, much as the serial digital computer did over forty years ago. Their pattern recognition capabilities will complement the available tools, methodologies, and techniques in an untold myriad of ways. The SSCG training method presented here, along with cooperative neural network signal recognition concepts, represents yet another step in the exploration of the potential of using neural networks to monitor and control a variety of aerospace and other related systems.

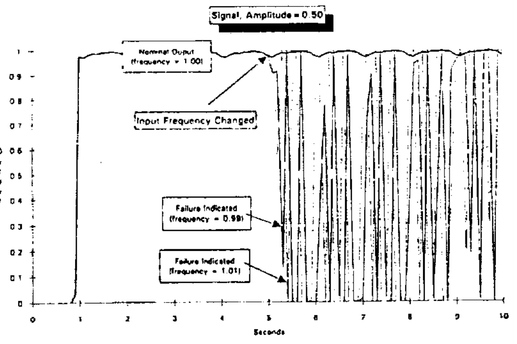
References

[1] Cybenko, G., "Approximations by Superpositions of a Sigmoidal Function," CSRD Report 856, UIUC Center for Supercomputing Research and Development, 1989, pp. 1-15.

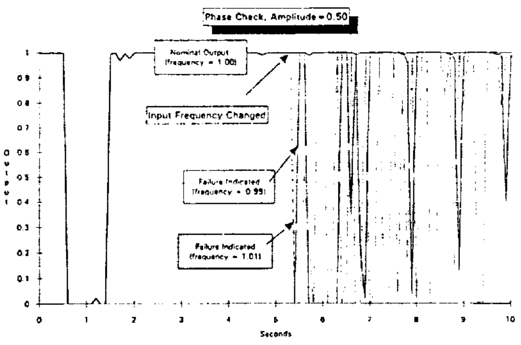
[2] Doyle, R. J., Berleant, D. Falcone, L. P., and Fayyad, U. M., "Selective Simulation and Selective Sensor Interpretation in Monitoring," AIAA 89-3101-CP, 1989, pp 859-870.

[3] Fahlman, S., "Faster Learning Variations on Back-Propagation: An Empirical Study," Proc. of the 1988 Connectionist Models Summer School at CMU, Morgan Kaufman Publishing, 1988, pp. 38-51.

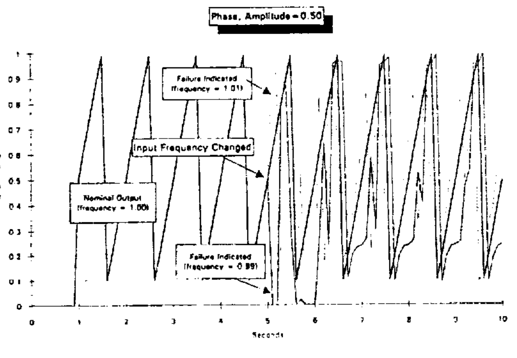
[4] Gargan Jr., R.A. and Kovarik Jr., V., "An Expert System for Mission Scheduling and Conflict Resolution," Proc. 2'nd Annual Workshop on Robotics and Expert Systems, Houston, TX, June 4-6, 1986.



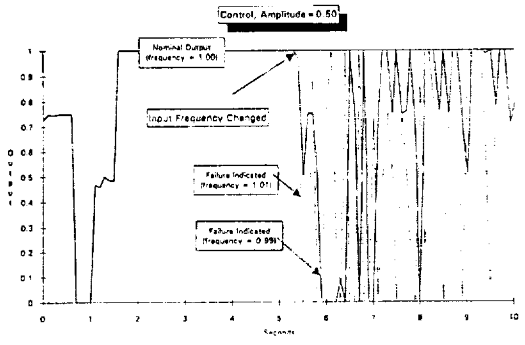
7a.



7c.



7b.



7d.

Figure 7. One Hz sine wave with induced frequency faults.

[5] Gill, P., Murray, W. ,and Wright, M., Practical Optimization, Academic Press, 1981, p. 150, 99, 141, 146.

[6] Goldberg, K. and Pearlmutter, B., "Using Backpropagation with Temporal Windows to Learn the Dynamics of the CMU Direct-Drive Arm II," Advances in Neural Information Processing 1, Morgan Kaufmann, 1989, pp. 356-363.

[7] Groundwater, E.H., Lembeck, M.F., and Sarsfield, L., "Development of An Expert Planning System for the Office of Space Science and Applications," 4'th Annual Conference on Artificial Intelligence for Space Applications, Huntsville, AL, November 1988.

[8] Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks are Universal Approximators," Neural Networks, Vol 2, 1989, pp. 359-366.

[9] Kohonen, T., "An Introduction to Neural Computing," Neural Networks, Vol 1, 1988, pp. 3-16.

[10] Lambe, J., Moopenn, A., and Thakoor, A.P., "Electronic Neural Networks," NASA Technical Support Package, NPO-16680/6175, 1988.

[11] Lembeck, Michael F., Design Methodology for a Neural Network-based Telemetry Monitor. University of Illinois at Urbana-Champaign, Dissertation, March 1991.

[12] Lippmann, R., "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, April, 1987, pp. 4-25.

[13] Lippmann, R., "Pattern Classification Using Neural Networks," IEEE Communications Magazine, November, 1989, pp. 47-64.

[14] Naidu, S. R., Zafirion, E., and McAvoy, T. J., "Use of Neural Networks for Sensor Failure Detection in a Control System," IEEE Control Systems Magazine, April, 1990, pp.49-55.

[15] Oren, S. S. and Spedicato, "Optimal Conditioning of Self-Scaling Variable Metric Algorithms," Math Programming, 10, 1976, p70-90.

[16] Pao, Y., Adaptive Pattern Recognition and Neural Networks, Addison-Wesley, Reading, MA, 1989.

[17] Rumelhart, D.E., McClelland, J.L., Parallel Distributed Processing, Vol 1, MIT Press, Cambridge, MA, 1986, p328, 330.

[18] Shanno, David F., "Conjugate Gradient Methods with Inexact Searches," Mathematics of Operations Research, Vol. 3, No. 3, August, 1978, pp244-256.

[19] Skapura, D. M., "A parallel Processor Designed for Artificial Neural Systems Simulation," AIAA 89-3090-CP, 1989, pp 798-803.

[20] Smith, R., "Structuring a Knowledge Base for Real-Time Diagnosis," Proc. of the International Symposium on Artificial Intelligence, Robotics, and Automation in Space, Kobe, Japan, 1990, pp. 197-200.