NEA/NSC-DOC (96)29
AUGUST 1996

SMORN-VII

REPORT
NEURAL NETWORK BENCHMARK

ANALYSIS RESULTS & FOLLOW-UP'96

Özer CIFTCIOGLU
Istanbul Technical University, ITU

and

Erdinç TÜRKCAN
Netherlands Energy Research Foundation, ECN

VOLUME I
(Revised, August 1998)

SMORN VII
A symposium on Nuclear Reactor
Surveillance and Diagnostics
13-19 June, 1995, Avignon (FRANCE)

NEURAL NETWORK BENCHMARK FOR SMORN-VII

ANALYSIS RESULTS & FOLLOW-UP'96

Özer CIFTCIOGLU
Istanbul Technical University, ITU
Electrical & Electronics Engineering Faculty, Istanbul, Turkey

and

Erdinç TÜRKCAN
Netherlands Energy Research Foundation, ECN
P.O. Box 1, 1755 ZG Petten, The Netherlands

SMORN VII
A symposium on Nuclear Reactor
Surveillance and Diagnostics
13-19 June, 1995, Avignon (FRANCE)

This work is accomplished under the auspices of OECD with an exclusive agreement between OECD and the first author affiliated with ITU.
Contract ref.: PERS/U3/96/509/AN/AMH.

NOTICE

©1996
All rights are at the discretion of OECD

The document is available at the web site:
http://www.nea.fr/html/science/rsd/

All correspondences should be addressed to:
Prof. Dr. Özer Ciftcioglu

Delft University of Technology
Informatica, BK 5e
Berlage weg 1
2629 CR Delft
The Netherlands

Tel : +31-15 2784485
Fax : +31-15 2784127
E-Mail: ciftciog@bk.tudelft.nl

CONTENTS

FOREWORD

NEURAL NETWORK BENCHMARK FOR SMORN-VII
ANALYSIS RESULTS: FOLLOW-UP'96

This work is a comprehensive final analysis report of the neural network benchmark (Benchmark-95) presented at **SMORN-VII**. The report contains the formal analysis of both the benchmark-data for the prescribed formal tasks and additionally some optional items placed in the benchmark exercise for optional further analyses. Altogether, the work carried out is a comprehensive study of the benchmark exercise using different approaches in neural network (NN) training and performing respective evaluations of the results. Finally, the benchmark analysis results reported by each participant are addressed, evaluated and documented in the report.

The work as a whole establishes some guidelines for neural network implementation especially for the applications in process system monitoring for safety and cost effective operation. In such applications the source of information is measurement data obtained from the process or plant sensors with random measurement errors. Therefore, the benchmark data are so prepared that they comprise the measurements obtained from an operating actual nuclear power plant to benefit the participants maximally as a return for their appreciable participation efforts.

The generalization capability is an important concept in the terminology of neural networks. In this respect, in the benchmark data distributed, *rinsing* and *stretch-out* operations are important modes for the evaluation of the neural network performance. Along this line, another important related issue is the concept of neural network sensitivity where the sensitivity should desirably be distributed evenly among the input-output signals. In the report, these topics are highlighted, investigated and the results are evaluated in the form of a research work beyond the prescribed formal analyses and foreseen tasks.

Considering this rather involved scope of Benchmark-95, the present study gives some indications about the state-of-the-art-functionality of the neural network technology for information processing and determines some guidelines for the effective neural network utilization especially having the industrial process monitoring, like a power plant for instance, in view.

SUMMARY

A neural network (NN) benchmarks designed for condition monitoring in nuclear reactors is described. As the neural network is a new technology arose in the last decade, its potentiality is getting recognized only recently and its utilization is already taking place in industrial applications. In this respect, the utilization of neural networks in nuclear reactors is also increasing for diverse applications. Among the most beneficial utilization one can refer to cost effective plant operation and plant monitoring and surveillance with enhanced safety which are due to the features inherently laying in the neural structure. The former is related to total productive maintenance including plant life extension. From the reactor noise analysis viewpoint, neural networks can be used for stochastic signal processing and spectral estimations as well as for processing operational information from gross plant signals. In the benchmark, which is shortly referred to as NN Benchmark-95, the latter, that is, information using gross plant signals from the detectors describing the operational status is considered because of both relatively easy processing of these signals with neural network and relatively easy evaluation of the results. The NN Benchmark-95 evaluation results have been globally disclosed at the symposium SMORN-VII and an extended summary have been presented. In this report, in addition to the comprehensive benchmark analysis results the benchmark data preparation is briefly described and the relevant tasks to be carried out by the participants are given for the completeness of the work. Also the prescribed tasks executed by the benchmark organizers are illustrated for guidance referring to both, the present participants and later probable embankments on the analysis of the benchmark data by prospective newcomers. This final report at hand is a comprehensive evaluation of the benchmark tasks analysis results submitted for formal participation and some elaboration on the optional tasks. Additionally, in the light of these considerations a follow-up of the NN Benchmark-95 is designed for prospective participants for the purpose of providing further insight into information processing by neural networks for enhanced utilization of this emerging technology in the nuclear technology.

1. INTRODUCTION

Although the neural networks, as a new information processing methodology, have appeared only in the last decade, the rapid growth of the research in this field is quite conspicuous. The applications almost in all engineering fields as well as in the soft sciences and other diverse areas resulted in a new technology known as Neural Network Technology today. The more recent growth of interest on the subject apparently to be caused by their promise to yield solutions that traditional approaches do not yield. Neural network (NN) is loosely articulated in the context of artificial intelligence, and therefore named as *artificial neural networks*, in the literature. It is claimed that it resembles the brain in some respects. In this context neural network can be defined as follows. A neural network is a distributed information processor where structural information can be stored and can be made available for use in later reference. It resembles the brain functions because the network through a learning process acquires knowledge and the information/knowledge stored is distributed in the form of connection weights of the network structure as brain does through synaptic weights connecting biological neurons to each other.

From the viewpoint of nuclear technology, in particular for power plant operation, the use of neural networks provides the following useful properties and capabilities.

1. ***Non-linearity***. The non-linearity is the inherent property of the neural network as it stores knowledge. By this property, the nonlinear relationships between the plant signals can easily be modeled.

2. ***Pattern recognition***. The most beneficial use of pattern recognition property in plant operation is presumably due to the input-output mapping performed by the network. The network learns from the examples by constructing an input-output mapping of the plant signals. Once this mapping is established through the learning process, responding to multi input excitations and observing deviations from this mapping scheme at the output of the network is a potential tool for monitoring and surveillance. Explicitly, a neural network may be designed so as to classify an input pattern as one of several predefined types of fault states of a power plant and afterwards for easy recognition of such a state in critical situations. These applications have demonstrated high performance. A more elaborated relevant application is for accident management [1].

3. ***Adaptivity***. By adaptivity, a neural network can operate in a non-stationary environment and it can be used for adaptive pattern classification like normal-abnormal status decision making in an operating plant. In particular, one can refer their ability to respond in real-time to the changing system-state descriptions provided by continuous sensor input. For a nuclear power plant where many sensors are used, the real-time response is a challenge to both human operators and expert systems. In this context, the integration of a neural network with an expert system provides neural network utilization with a substantial potentiality for fault diagnosis. In this case, monitoring system thus formed gains the functionality of being an operating support system as the decisions made are based on the power plant knowledge base of the expert system.

4. ***Fault tolerance***. In hardware form, fault tolerance of a neural network refers to the performance of the network that it is degraded gracefully under adverse operating

conditions. In software form, the same refers to the performance of the network where a neural network has the ability to recognize patterns, even the information making up these patterns is noisy or incomplete. Because of this very feature, the adaptive neural networks are highly suited for fault diagnosis and control and risk evaluation in nuclear power plant environments.

Neural network is a generic name for a number of different constructions where artificial neural nodes are connected among themselves. The connections are called as (synaptic) weights. Presumably, feed-forward neural networks are the most popular types among others. Many reported works with these types of networks indicated the recognition of neural network technology today. Although the reported outcomes showed the desire for effective utilization and exploitation of the potentiality of this technology, globally, there is no consensus about the exact form of forming the network structure and the training procedures. In other words, for the same data and network type, there are many different choices for forming the network structure as well as for its training with different input parameters apart from a number of different choices of input data reduction and/or preparation. Therefore the comparison of different approaches would be very informative to get insight into understanding the nature of differences among the various neural networks' performances and establish some general rules to be observed for enhanced utilization of neural networks.

From the above described viewpoint it was decided to establish a benchmark exercise for investigation of potential utilization of neural networks in condition monitoring in power plants and try to establish some basic common understanding for the application of this emerging technology. By means of this some standard rules of utilization or recommendations could be concluded and everyone working actively in this field could benefit. Hence, it is anticipated that the exercise would reveal or shade some light on the applicability of neural network technology in nuclear industry for monitoring.

To this end, a set of real plant data for condition monitoring exercise from the plant sensory signals is prepared having in mind that the condition monitoring is the common feature of all modern maintenance technologies and in this respect the data at hand would be of general concern. The data were distributed to the participants by OECD-NEA to carry out the tasks of the benchmark exercise and reporting as feedback. They are asked to report their results in a prescribed form for precise comparisons among the reported results. This would also provide the benchmark organizers with the possibility of reproducibility of reported results for the purpose of probable detailed studies.

With reference to the scope of SMORN VII, neural networks formed a new technology that can be closely associated with the reactor noise analysis studies and the associated technologies. Because of this newly emerging feature and concerning their actual utilization in industry, neural networks are not widely known to the community of SMORN. Therefore to disseminate the benchmark results effectively and benefit the community from this comprehensive work and, at the same time, to encourage the prospective new participation in a similar benchmark framework, some associated essential information in a limited form will be presented. For that purpose, in the following section some basics of neural networks will be given. This will provide a brief information about the field in order to benefit those who are not directly interested in neural networks or did not participate in the present benchmark, i.e., NN Benchmark-95, but they are interested in the benchmark results as well as

in getting to know about the neural networks and their utilization for power plant monitoring. As the field of neural networks is immense, even such a brief description of basics would be quite voluminous and would be out of the scope of this report. Therefore, in this description especially reference is made to the reported results from the participants. Next to the feed-forward neural networks, the description will especially highlight the topics dealt with in the participants' reports as well as dealt with in this formal analysis report of the benchmark at hand. This might be of help to assist the reader for better comprehension of the participants' reports as well as the results evaluated and conclusions reported here. The latter includes also recommendations in the form of proposals for a follow-up of the NN Benchmark-95.

The organization of this report is as follows:

The report preliminarily introduces neural network as a tool for information processing having the benchmark data analysis and the related tasks in view. In this respect, Section 2 deals with neural networks. The first subsection of chapter 2, deals with the essentials of the well-known standard feed-forward multi-layer feed-forward (MLP) neural networks. Following this, the close associate of the MLP is the radial-basis function networks (RBF) is outlined. MLP and RBF networks have the same neural structure and they can be used at the same applications as efficient function approximators. Although their structures are the same, their training and operation differ considering further details and merits. In connection with the training of RBF networks, the self-organizing neural networks are briefly introduced and this concludes the essentials of the neural network structures of interest in this work. The second subsection deals with the learning algorithms being associated with the neural networks considered in the preceding subsection. Basically, the learning algorithms that are alternatively termed as *training algorithms* play important role on the performance of the associated network. This is simply due to fact that the learning process is used to carry out the storage of in advance acquired and processed information at hand in to the neural network. The knowledge acquisition capacity of the network is, therefore, mainly dependent on this process as well as its inherent neural structure. In view of this importance, appropriate learning methods and the related algorithms for the analysis of the benchmark data are briefly described. Preliminarily this description is divided into two major groups. In the first group, supervised learning algorithms and initially, the back-propagation (BP) algorithm is outlined, since this is perhaps the most popular algorithm due to as it is relatively much easier to follow and implement. Following the BP algorithm, in the same group, nonlinear unconstrained parameter optimization methods are outlined. Among these, conjugate-gradient, quasi Newton methods are the most appropriates ones for the intended tasks in this benchmark exercise. With reference to these algorithms, the BP algorithm is essentially also a nonlinear algorithm and similar to the one known as steepest-descent. However, due to the approximate-gradient involvement during the convergence process, it becomes a *stochastic approximation* in the optimization process. Considering this form, another advanced stochastic approximation methods are the *recursive-least-squares* (RLS) and *Kalman filtering* approaches for supervised learning. Both methods and the related algorithms have concluded that subsection. In the second group of learning, an unsupervised learning algorithm is considered. Although there are a number of such algorithms, from the benchmark goals viewpoint, the algorithm subject to consideration is the Khonen's self-organizing-feature-map (SOFM) algorithm due to its suitability for use in RBF network training. In the RBF training, the self-organizing algorithm constitutes one of the four alternative training algorithms in use.

The last subsection of Section 2 considers the sensitivity in neural networks. Although, *sensitivity* is well-defined concept for engineering systems, the definition does not have immediate physical interpretation in the case of neural network systems, as a neural network is a computational model developed without having recourse to a physical or other actual model. However, in this very context, sensitivity attribution has very deep implications and therefore an appropriate interpretation would help greatly for enhanced and effective utilization of neural networks. In this subsection these issues are addressed.

In Section 3, the analysis of the benchmark results reported is presented. Nine participant-reports are analyzed and assessed in perspective. Some major issues are highlighted and some immediate explanatory statements are included.

In Section 4, further analyses results with the benchmark data are presented. The motivation is the demonstration of the previous information given about neural networks and their enhanced and effective use. These analyses are primarily directed to those by whom, the basics of such utilization for power plant monitoring and fault detection having been satisfactorily comprehended, they wish to continue along with the neural network technology. Together with the analyses results, the relevant peculiarities of the data are consistently explained and interesting features of power plant monitoring with neural networks are pointed out.

In Section 5, a follow-up of the existing benchmark exercise prepared for SMORN-VII is considered and some follow-up schemes are proposed. These proposals aim to the furtherance of the application of neural network technology to power plant operation for plant safety and cost effective operation. To achieve these goals reactor noise studies should desirably be associated with the neural network technology or vice versa. In this context, among others, a proposal is also devised for stochastic signal processing by neural networks where dynamic signal modeling and analysis by neural networks rather than model validation by gross plant signals, is considered. Presumably, such utilization would be of most concern to the community interested in reactor noise.

In the last section (Section 6) the conclusions drawn from the analysis are presented and thereby the comprehensive final evaluation report of the benchmark is thus concluded.

The report is appended with the NN Benchmark-95 extended summary report, formal benchmark questionnaire and the participants' submissions for the sake of completeness of this work which can *eventually* be considered as a complete documentation of the present benchmark execution.

2. NEURAL NETWORKS

2.1. Network Structures

Feed-Forward Multi-layer Perceptron Neural Networks

Neural network is a data processing system consisting of a number of simple, highly interconnected processing elements in a input/output architecture. Since a processing element models a biological neuron, it is also called neuron in this context. A neural network is generally structured with a set of a prescribed number of elementary processing units arranged in several layers. Following an input layer which serves merely for data introduction, at the first neuronal layer that is called first hidden layer, each neuron receives at its inputs, from external information sources. At the subsequent layers that are called subsequent hidden layers, each neuron receives at its inputs the outputs of the preceding layer through weighted connections that are called synaptic weights. In particular, the last neuronal layer is called output layer. The very first layer that is, input layer where no neuron exists, external information is coupled to the network. Each hidden layer receives the information provided by the immediate lower level neurons and sends the processed information directly to the upper layer. The output-layer delivers the final processed information for use in the intended application.

Each processing element is connected to the preceding layer by means of weights as the information incoming to the element are weighted and summed in the processing element together with a weighted constant bias called threshold. The nonlinear multi-input and single-output transfer function at the processing element is called activation or logistic function and there are several choices for it. The mostly used function is a sigmoid function that is represented by

$$y_j(n) = \frac{1}{1 + \exp(-v_j(n))} \qquad -\infty < v_j(n) < \infty$$

with the definition of

$$v_j(n) = \sum_{i=1}^{p} w_{ji}(n)\, y_j(n) + \vartheta_j$$

Above, n is the iteration number (i.e., presentation of the n-th training pattern); j, index number for a neuron; p, is the total number of inputs applied to neuron j; w_{ji} is the synaptic weight connecting neuron i to neuron j; $y_i(n)$ is the input signal of neuron j or in other terms, the function signal appearing at the output of neuron i; ϑ, is externally applied constant input called threshold. Another type of sigmoidal non-linearity is the hyperbolic tangent, which is anti-symmetric with respect to the origin and for which the amplitude of the output lies inside the range $-1 \leq y_i \leq +1$.

This final configuration, i.e., a processing element with its synaptic weights and the threshold and a coupled activation function is called perceptron. This very structure in progressive form provides the structure known as multi layer perceptron (MLP) network. Here the input signal propagates through the network in a forward direction, on a layer-by-layer basis. The most popular form of MLP network is feed-forward neural network. A typical *feed-forward neural network* structure and a neural processing element are schematically shown in Fig.1.

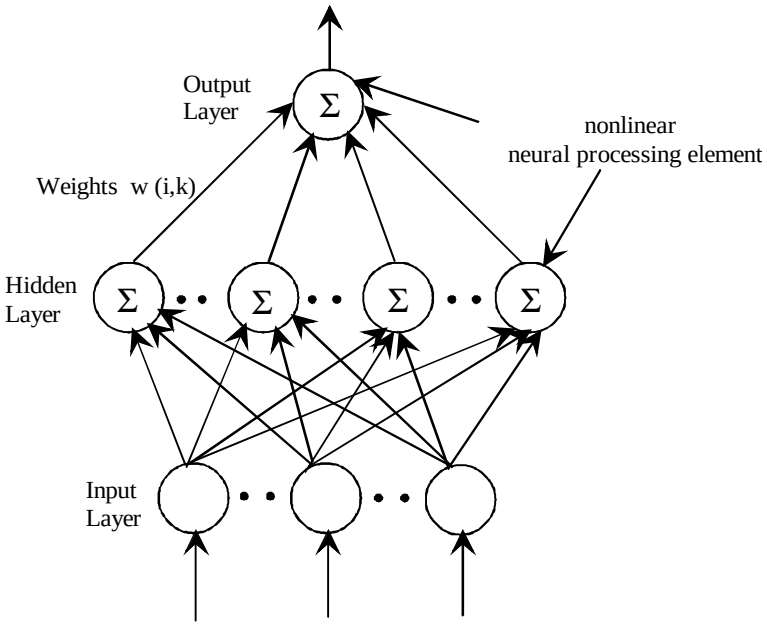

Fig.1. Schematic representation of a feed-forward neural network

Radial-Basis Function Networks

Another neural network structure similar to the feed-forward neural networks is the Radial-Basis Function Networks. The construction of a radial-basis function (RBF) network in its most basic form comprises three entirely different layers. The input layer that is made up of sensory source nodes. The second layer is a hidden layer of radial base functions. The output layer that gives the mapping of the input pattern. The transformation from the input space to the hidden-unit space is nonlinear, whereas the transformation from the hidden-unit space to the output space is linear. Architecture of a RBF network is shown in Fig.2.

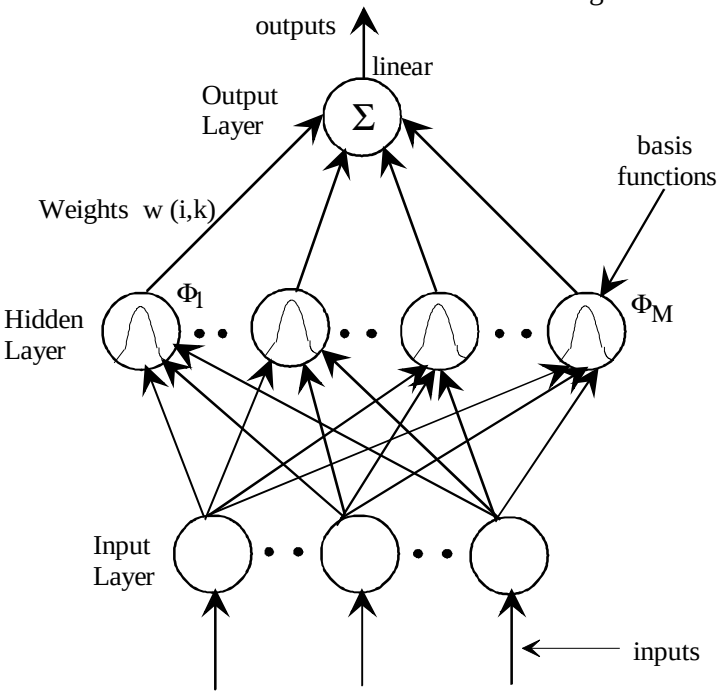

Fig.2: Architecture of a radial-basis function network

Radial-basis function network and the multi-layer perceptrons are examples of nonlinear layered feed-forward networks. They are both universal approximators. It is therefore always possible to establish a RBF network capable of being accurately alternative to a specified multi-layer perceptron, or vice versa. The differences can be summarized as follows.
1. An RBF network has a single hidden layer, whereas MLP network can have one or more hidden layers
2. The argument of the activation function of each hidden unit in a RBF network computes the Euclidean norm between the input vector and the center of that unit. On the other hand, the activation function of each unit in a MLP network computes the inner product of the input vector and the synaptic weight vector of that unit.

The mapping by the RBF network has the form

$$y_k(\mathbf{x}) = \sum_{j=1}^{M} w_{kj}\, \phi_j(\mathbf{x}) + w_{ko}$$

where M is the number of hidden layer nodes; k, is the index parameter for the output nodes; ϕ_j is the basis function. The biases can be included into the summation by including an extra basis function ϕ_o whose activation is set to 1. This is similar to the case in MLP networks. For the case of Gaussian basis functions we have

$$\phi_j(\mathbf{x}) = \exp\left(\frac{\|\mathbf{x} - \mathbf{m}_j\|^2}{2\sigma_j^2}\right)$$

where $\mathbf{x}$ is the input vector with elements x_i ; $\mathbf{m}_j$, is the vector determining the center of the basis function ϕ_j and has elements μ_{ji} . The Gaussian basis functions are not normalized.

Self-Organizing Neural Networks

Neural networks can learn from the environment, and through learning they improve their performance. Multi-layer perceptron networks and radial-basis function networks learn by an external teacher which provides the network with a set of targets of interest as this will be termed as supervised learning, and will be explained shortly afterwards. The targets may take the form of a desired input-output mapping that the algorithm is required to approximate. In self-organizing systems the network weights are determined by considering the input information only, without recourse to an external teacher.

The structure of a self-organizing system may take on a variety of different forms. A most popular form consists of an input (source) layer and an output (representation) layer, with feed-forward connections from input to output and lateral connections between neurons in the output layer. In another form, it may be a feed-forward network with multiple layers, in which the self-organization proceeds on a layer-by-layer basis to develop a final configuration concerning the weights. The output neurons are usually arranged in a one- or two-dimensional lattice, i.e., a topology that ensures that each neuron has a set of neighbors. An example of such a structure is known as Khonen model self-organizing network and shown in Fig.3. The Khonen model has received relatively much more attention in implementation. This is because the model possesses certain properties. The associated learning algorithm for this network is called self-organizing feature map (SOFM) which will briefly be described in the following section.

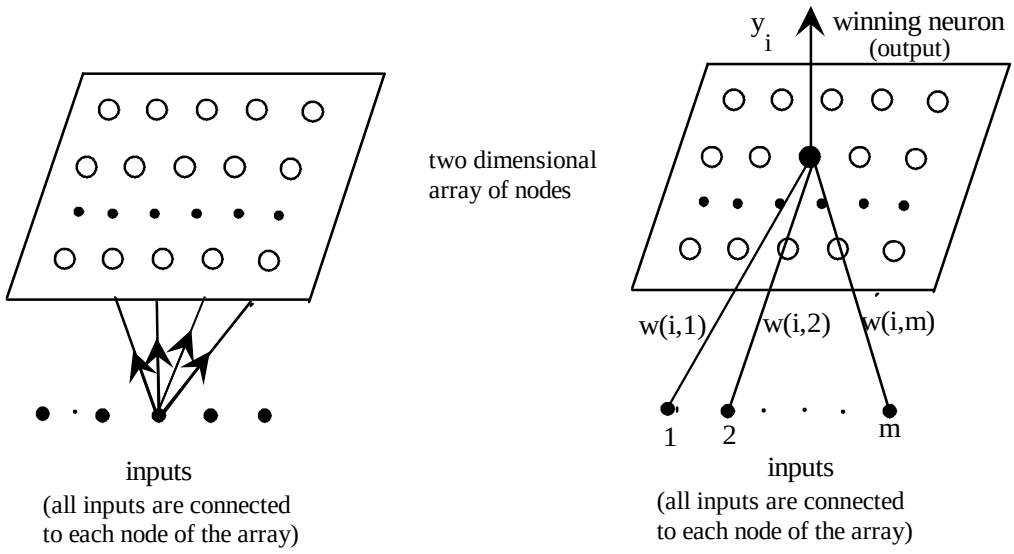

Fig.3. Schematic representation of Khonen self-organized neural network

2.2. Learning Algorithms

In relation to the benchmark exercise and the related tasks, the suitable feed-forward neural network structures, i.e., multi-layer perceptron (MLP) and radial-basis function (RBF) networks are introduced above. In relation to the latter, a self-organizing network structure is also presented. In this subsection the learning algorithms subject to training these networks will be given.

There are several different kinds of learning commonly used with neural networks. Perhaps the most common is supervised learning in which a stimulus is presented at the input layer of the network and the output from the output layer is sent to a system. The system compares it with a desired output and then uses a corrective or learning algorithm to convert the error formed by the difference into an adjustment of the weighting coefficients that control the inputs to the various processing elements. In a typical situation, the initial weighting functions are set randomly and then subjected to incremental changes determined by the learning algorithm. When another input is again applied to the input layer, it produces an output that is again compared with the desired output to produce a second error signal. This iterative process continues until the output of the neural network is virtually equal to the desired output. At that point the network is said *trained*. Further, through the various learning algorithms, the network gradually configures itself to achieve the desired input/output relationship called ‘mapping’.

The counterpart of the supervised learning is the unsupervised learning algorithm where only the input stimuli are applied to the input neuronal layer of the network. Due to single neuronal layer this structure is called single layer network in contrast with multi-layer network subject to supervised learning. The network then organizes itself internally so that each hidden processing element responds strongly to a different set of input stimuli. As result of learning or *training*, the sets of input stimuli represent clusters in the input space. Referring to this peculiarity of the learning process this network is called self-organizing neural network. Generally the neuronal layer is one or two-dimensional. A typical *self-organizing neural- network* the form of which is referred to as Kohonen network is shown in Fig.3.

2.2.1. Supervised Learning Algorithms

Back-propagation Algorithm

Feed-forward neural networks have received presumably the most intensive attention and put into operation in actual implementations due to their simplicity with respect to both training and operation together with their success in diverse applications. Nuclear industry is one of the application areas where they have a wide range of application. The most popular algorithm for training is known as the *error back-propagation algorithm*. This algorithm is a stochastic gradient descent algorithm that is a counterpart of the well-known linear least-mean-square (LMS) algorithm. The error back-propagation process is accomplished in two passes, namely forward and backward, through the different layers of the network. In the forward pass an input vector is applied to the input nodes of the network. The effect is propagated forward until the output. As the algorithm is a supervised one, the error between the actual output and the desired output is used to adjust the weights according to the error-correction rule. The error signal thus propagates backward through the network. The synaptic weights are adjusted so as to make the actual response of the network more closer to the desired response. There are two basic modes for this process. These modes are *pattern* and *batch* modes of training. In the pattern-based back-propagation learning, weight updating is performed after the presentation of each training example sequentially. In the batch mode of back-propagation learning, weight updating is performed after the presentation of all the training examples that constitutes the whole set of the patterns called an *epoch*. For weight update the error function gradient is evaluated and the weights updated using

$$\Delta \mathbf{w} = - \eta \nabla E$$

where $\Delta \mathbf{w}$ is the weight update; ∇E is the gradient of the error function. η is called the learning rate and provided its value is sufficiently small, error function E the value of which desired to be minimum will decrease at each successive step.

Although the back-propagation algorithm is relatively simple to implement it suffers because of slow convergence and local trappings during the convergence. Therefore, a number of variants of the basic algorithm is developed along two different modes of back-propagation training. Among the important back-propagation variants mention can be made to modified gradient descent using momentum term addition to the error term and iteration dependent learning-rate changing known as *delta-bar* learning rule. Both variants are devised for the accelerated converge during training avoiding the local minima. The modified gradient descent formula is given by

$$\Delta \mathbf{w} = -\eta \nabla E(n) + \mu \Delta \mathbf{w} (n-1)$$

where μ is called the momentum parameter. The inclusion of momentum generally leads to a significant improvement in the performance of gradient descent. The same applies also to the delta-bar learning. Nevertheless, in both cases the algorithm remains relatively inefficient. This motivates one to seek for advanced function minimization methods which directly leads to the exploitation of appropriate advanced mathematical methods called optimization.

Application of BP algorithm for power plant monitoring is described before [2,3].

Parameter Optimization Algorithms

As the supervised neural network training can be viewed as a special case of function minimization, advanced nonlinear optimization algorithms can be applied for that purpose. Among these grid search, line search, gradient search, gradient expansion, conjugate-gradient, quasi-Newton can be referred. These methods all have substantial merits subject to application. The common features of the optimization methods are the followings.

1. Entire set of weights are adjusted once instead of sequential adjustments
2. The entire set of patterns to the network is introduced and the sum of all squared errors from the patters is used as the objective function.

During the iterative strategy of an optimization method, an appropriate search direction in the space of the weights is selected. In that direction a distance is moved so that the value of the objective function is improved when the new values of the weights are introduced into it. For efficient i.e., fast and effective i.e., avoiding local minima, neural network training, conjugate-gradient and quasi-Newton are of major concern. These are briefly described below.

An essential degree of insight into the optimization problem, and into the various techniques for solving it, can be obtained by considering a local quadratic approximation to the error function. Let us consider the Taylor expansion of E(**w**) around some point $\mathbf{w}^*$ in the weight space

$$E(\mathbf{w}) = E(\mathbf{w}^*)+(\mathbf{w}-\mathbf{w}^*)\,\mathbf{b} + \tfrac{1}{2}\,(\mathbf{w}-\mathbf{w}^*)^T H(\mathbf{w}-\mathbf{w}^*)$$

where **b** is is the gradient of E evaluated at **w***

$$\mathbf{b} = \nabla E$$

and the Hessian matrix **H** is defined by

$$(\mathbf{H})_{ij} = \frac{\partial E}{\partial w_i\, \partial w_j} \Big|_{\mathbf{w}^*}$$

The local approximation for the gradient is given by

$$\nabla E = \mathbf{b} + \mathbf{H}\,(\mathbf{w}-\mathbf{w}^*)$$

For points **w** which are close to **w***, the quadratic approximation is reasonably valid for the error and its gradient. Consider the special case of a local quadratic approximation around a point $\mathbf{w}^*$ which is a minimum of the error function. In this case there is no linear term, since $\nabla E=0$ at $\mathbf{w}^*$, and therefore the error function becomes

$$E(\mathbf{w}) = E(\mathbf{w}^*) + \tfrac{1}{2}\,(\mathbf{w}-\mathbf{w}^*)\,\mathbf{H}\,(\mathbf{w}-\mathbf{w}^*)$$

Consider the eigenvalue equation for the Hessian matrix

$$\mathbf{H}\,\mathbf{u}_i = \mathbf{l}_i\,\mathbf{u}_i$$

where the eigenvectors $\mathbf{u}_i$ form a complete orthonormal set, i.e **.,**

$$\mathbf{u}_i^T\,\mathbf{u}_j = \delta_{ij}$$

We expand $(\mathbf{w} - \mathbf{w}^*)$ as a linear combination of the eigenvectors in the form

$$\mathbf{w} - \mathbf{w}^* = \sum_i \alpha_i \mathbf{u}_i$$

so that the substitution yields the error function of the form

$$E(\mathbf{w}) = E(\mathbf{w}^*) + \tfrac{1}{2} \sum_i \lambda_i \alpha_i^2$$

In this procedure the coordinate system of $\mathbf{w}$ is transformed the another one in which the origin is translated to, and the axes are rotated to align with the eigenvectors through the orthogonal matrix whose columns are $\mathbf{u}$.

Since the eigenvectors $[\mathbf{u}_i]$ form a complete set, an arbitrary vector $\mathbf{v}$ can be written

$$\mathbf{v} = \sum_i \beta_i \mathbf{u}_i$$

and therefore we write

$$\mathbf{v}^T \mathbf{H} \mathbf{v} = \sum_i \beta_i^2 \lambda_i$$

and so $\mathbf{H}$ will be positive definite i.e., $\mathbf{v}^T \mathbf{H} \mathbf{v} > 0$ for all $\mathbf{v}$, if all of its eigenvalues are positive. In the new coordinate system whose basis vectors are given by the eigenvectors $[\mathbf{u}_i]$, the contours of constant E are ellipses centered on the origin, whose axes are aligned with the eigenvectors and whose lengths are inversely proportional to the square roots of the eigenvalues. This is illustrated in Fig.4. For a one-dimensional weight space, a stationary point $\mathbf{w}^*$ will be minimum if

$$\frac{\partial E}{\partial \mathbf{w}} \Big|_{\mathbf{w}^*} > 0$$

The corresponding result is that the Hessian matrix, evaluated at $\mathbf{w}^*$ is positive definite.

Now, considering the quadratic error function E($\mathbf{w}$) the local gradient of this error function is given by

$$\mathbf{g}(\mathbf{w}) = \mathbf{b} + \mathbf{H}\,\mathbf{w}$$

and the error function E(w) is minimized at the point $\mathbf{w}^*$ given the preceding equation, by

$$\mathbf{b} + \mathbf{H}\,\mathbf{w}^* = 0$$

Suppose that at step n the current weight vector is $\mathbf{w}(n)$ and we seek a particular search direction $\mathbf{d}$ along which

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \lambda(n)\,\mathbf{d}(n)$$

and the parameter $\lambda(n)$ is chosen to minimize

$$E(\lambda) = E(\mathbf{w}(n) + \lambda\,\mathbf{d}(n)) .$$

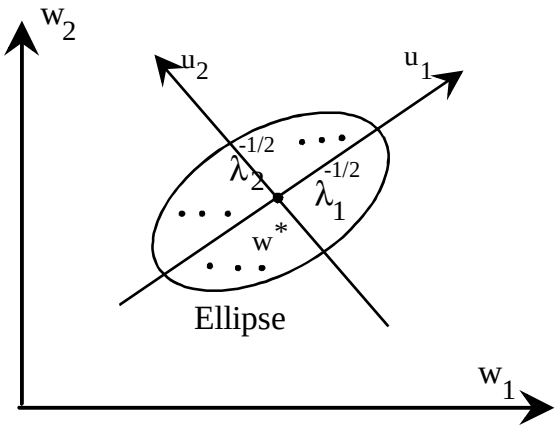

Fig.4 : Illustration of the coordinate transformation by means of principal component analy sis

This requires that

$$\frac{\partial}{\partial \lambda} E(\mathbf{w}(n) + \lambda\, \mathbf{d}(n)) = 0$$

which gives

$$\mathbf{g}^T (n+1)\ \mathbf{d}(n) = 0$$

that is, the gradient at the new minimum is orthogonal to the previous search direction where $\mathbf{g} = \nabla E$. In the successive search, direction $\mathbf{d}(n)$ is selected such that, at each step the component of the gradient in the previous search direction, is not altered so that minimum at that direction is maintained. In particular, at the point $\mathbf{w}(n+1)$, we have

$$\mathbf{g}^T(\mathbf{w}(n+1))\ \mathbf{d}(n) = 0$$

We now choose the next search direction $\mathbf{d}(n+1)$ such that, along this new direction

$$\mathbf{g}^T(\mathbf{w}(n+1) + \lambda\, \mathbf{d}(n+1))\ \mathbf{d}(n) = 0$$

holds. If, we expand the preceding expression to first order in λ, we obtain

$$\mathbf{d}^T(n+1)\ \mathbf{H}\ \mathbf{d}(n) = 0$$

where $\mathbf{H}$ is the Hessian matrix evaluated at the point $\mathbf{w}(n+1)$. If the error surface is quadratic, this relation holds for arbitrary values of λ. Search directions complying with the requirement stated above are said to be conjugate. Approaching to minimum in this way forms the conjugate gradient algorithm. We can find a set of P vectors where P is the dimensionality of the weight space, which are mutually conjugate with respect to $\mathbf{H}$, yielding

$$\mathbf{d}_j\ \mathbf{H}\ \mathbf{d}_i = 0 \qquad j \neq i$$

This implies that these vectors are linearly independent provided $\mathbf{H}$ is positive definite. Hence, such vectors constitute a complete basis set in weight space. Assume we are at some point $\mathbf{w_1}$, and we aim to reach to the minimum $\mathbf{w}^*$ of the error function.

The difference between the vectors $\mathbf{w}_1$ and $\mathbf{w}^*$ can be written as a linear combination of the conjugate direction vectors in the form

$$\mathbf{w}^* - \mathbf{w}_1 = \sum_{i=1}^{P} \alpha_I \mathbf{d}_i$$

This equation can be re-written recursively as

$$\mathbf{w}_{j+1} = \mathbf{w}_j + \alpha_j \mathbf{d}_j$$

where

$$\mathbf{w}_j = \mathbf{w}_1 + \sum_{i=1}^{j-1} \alpha_i \mathbf{d}_i$$

In the successive steps, the step length is controlled by the parameter α_j. From the above equations the explicit solution for these parameters is found to be

$$\alpha_j = - \frac{\mathbf{d}_j^T (\mathbf{b} + \mathbf{H} \mathbf{w}_1)}{\mathbf{d}_j^T \mathbf{H} \mathbf{d}_j}$$

or alternatively

$$\alpha_j = - \frac{\mathbf{d}_j^T \mathbf{g}_j}{\mathbf{d}_j^T \mathbf{H} \mathbf{d}_j}$$

Following the iterative procedure

$$\mathbf{w}_j = \mathbf{w}_1 + \sum_{i=1}^{j-1} \alpha_i \mathbf{d}_i$$

the gradient vector $\mathbf{i}$ at the j-th step is orthogonal to all previous conjugate directions. This implies that after P steps the components of the gradient along all directions have been made zero, that is the minimum is reached with the assumption the error surface is quadratic.

The construction of the set of mutually conjugate directions is accomplished as follows. Initially, the search direction is chosen to be the steepest descent, i.e., negative gradient $\mathbf{d}_1 = -\mathbf{g}_1$. Then, each successive direction is chosen to be a linear combination of the current gradient and the previous search direction

$$\mathbf{d}_{j+1} = - \mathbf{g}_{j+1} + \beta_j \mathbf{d}_j$$

where β is determined substituting this equation into the equation $\mathbf{d}_j \mathbf{H} \mathbf{d}_i = 0$ that it gives

$$\beta_j = \frac{\mathbf{g}_{j+1}^T \mathbf{H} \mathbf{d}_j}{\mathbf{d}_j^T \mathbf{H} \mathbf{d}_j}$$

It follows that by means of the successive use of the iterative search direction expression a set of P mutually conjugate directions are generated. Also at the same time $\mathbf{d}_k$ is given by the linear combination of all previous gradient vectors

$$\mathbf{d_k} = - \mathbf{g}_k + \sum_{m=1}^{k-1} a_m \mathbf{g_m}$$

Since $\mathbf{d_k}^T \mathbf{g_j} = 0$ for all $k < j \leq P$, we have

$$\mathbf{g_k}^T \mathbf{g_j} = \sum_{m=1}^{k-1} a_m \mathbf{g_m}^T \mathbf{g_j} \qquad \text{for all} \qquad k < j \leq P$$

Since the initial search direction is just $\mathbf{d}_1 = - \mathbf{g_1}$, from the equation

$$\mathbf{g}^T (n+1)\mathbf{d}(n) = 0$$

above $\mathbf{g_1}^T \mathbf{g_j} = 0$

so that the gradient at step j is orthogonal to the initial gradient. Further, from here by induction we find that the current gradient is orthogonal to all previous gradients

$$\mathbf{g_k}^T \mathbf{g_j} = 0 \quad \text{for all} \quad k < j \leq P$$

The method described above can be summarized as follows. Starting from a randomly chosen point **w** in the weight space, successive conjugate directions are constructed using the parameter β_j. At each step, the weight vector is incremented along the corresponding direction using the parameters α_j. The method finds the minimum of a general quadratic error function in at most P steps.

The Conjugate-Gradient Algorithm

In the conjugate-gradient algorithm, the step length is determined by α_j and the search direction is determined by β_j.. These coefficients are dependent on the Hessian matrix **H**. However, for the efficiency of the algorithm the computation of the Hessian is desirable avoided. For this reason for β_j different choices are possible which are obtained from the relationships given above. These are

$$\beta_j = \frac{\mathbf{g_{j+1}}^T (\mathbf{g_{j+1}} - \mathbf{g_j})}{\mathbf{d_j T} (\mathbf{g_{j+1}} - \mathbf{g_j})}$$

or

$$\beta_j = \frac{\mathbf{g}_{j+1} (\mathbf{g_{j+1}} - \mathbf{g_j})}{\mathbf{g_j}^T \mathbf{g_j}}$$

or alternatively

$$\beta_j = \frac{\mathbf{g}_{j+1}^T \mathbf{g}_{j+1}}{\mathbf{g}_j^T \mathbf{g}_j}$$

These expressions are the equivalent for a quadratic function subject to minimization. In the same way as β_j is calculated without Hessian information, the coefficient α_j is calculated likewise by

$$\alpha_j = - \frac{\mathbf{d}_j^T \mathbf{g}_j}{\mathbf{d}_j^T \mathbf{H} \mathbf{d}_j}$$

This is the expression indicating a line minimization along the search direction $\mathbf{d}_j$. The steps of the algorithm can be summarized as follows:

1- Choose an initial weight vector $\mathbf{w}_o$
2- Evaluate the gradient vector $\mathbf{g}_o$, and set the initial search direction $d_o = - \mathbf{g}_o$
3- At step j, minimize $E(\mathbf{w}_j + \alpha \mathbf{d}_j)$ with respect to α to obtain $\mathbf{w}_{j+1} = \mathbf{w}_j + \alpha_{min} \mathbf{d}_j$
4- Check if the minimum is reached
5- Evaluate the new gradient vector $\mathbf{g}_{j+1}$
6- Evaluate the new search direction by computing β_j
7- Set j=j+1 and go to step 3

The Quasi-Newton Method

Using the local quadratic approximation, we can obtain directly an expression for the location of the minimum of the error function. From the quadratic error function

$$E(\mathbf{w}) = E(\mathbf{w}) +(\mathbf{w}-\mathbf{w}^{\wedge}) \mathbf{b} + ½ \ (\mathbf{w}-\mathbf{w}^{\wedge})^T H(\mathbf{w}-\mathbf{w}^{\wedge})$$

the gradient at any point $\mathbf{w}^*$ is given by the Newton formula

$$\mathbf{g} = \nabla E = \mathbf{H} (\mathbf{w} - \mathbf{w}^*)$$

so that the weight vector w corresponding to the minimum is computed to be

$$\mathbf{w}^* = \mathbf{w} - \mathbf{H}^{-1} \mathbf{g}$$

Since the quadratic approximation used is not exact it would be necessary to apply the preceding weight updating iterative, with the inverse Hessian being repeatedly computed at each new search point. To avoid this lengthy operations, alternatively, a sequence of matrices $\mathbf{G}(n)$ are generated which represent $\mathbf{H}^{-1}$ with increasing accuracy, using the information on the first derivatives. To ensure that the Hessian is positive definite a special update procedure is developed.

From the Newton formula, the weight vectors at steps n and n+1 are related to the corresponding gradients by

$$\mathbf{w}^{(n+1)} - \mathbf{w}^{(n)} = - \mathbf{H}^{-1} (\mathbf{g}^{(n+1)} - \mathbf{g}^{(n)})$$

which is known as the Quasi-Newton condition. Assuming H is constant in the neighborhood of a minimum, we can write

$$\Delta \mathbf{w_o} - \Delta \mathbf{w} = -\mathbf{H}^{-1} \mathbf{g}^{n+1}$$

Above, $\Delta \mathbf{w_o}$ is the weight increase required to reach the minimum, and $\Delta \mathbf{w}$ is the actual weight increase. Since **H** is not constant, $\Delta \mathbf{w}$ above cannot be directly calculated. Therefore for the evaluation of $\mathbf{H}^{-1}$, a symmetric non-negative definite matrix **G** is considered which is in particular is to be positive if there are no constraints on the variables and might initially be a unit matrix. This matrix is modified after the i-th iteration using the information gained by moving down the direction

$$\Delta \mathbf{w}_i = -\mathbf{H}_i \mathbf{g}_i \qquad i= 1,2,...,N$$

in accordance with the equation before the preceding one. This direction points towards the minimum of the error function. The commonly used update formula is the Davidson-Fletcher-Power algorithm where, while moving down to the minimum, the line minimization is performed, as it is the case for conjugate-gradient method. Therefore the method is called *variable-metric minimization*. The basic procedure for the quasi-Newton method is as follows.

1- A starting point is selected

2- define $\mathbf{s}_I = \alpha_I \Delta \mathbf{w}_i$

3- set the weight values as $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{s}_I \qquad i=1,2,.....,N$

4- evaluate $E(\mathbf{w}_{k+1})$ and $\mathbf{g}_{k+1}$

5- define $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$

$$\mathbf{A}_k = \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{s}_k \mathbf{y}_k} \quad , \quad \mathbf{B}_k = - \frac{\mathbf{G}_k \mathbf{y}_k \mathbf{y}_k^T \mathbf{G}_k}{\mathbf{y}_k^T \mathbf{G} \mathbf{y}_k}$$

and evaluate the matrix G_{k+1} by setting

$$\mathbf{G}_{k+1} = \mathbf{G}_k + \mathbf{A}_k + \mathbf{B}_k$$

6- Set k=k+1 and start from step 2, repeat the iteration until the minimum is reached.

The advantages of quasi-Newton (variable-metric) method can be stated as follows.
- Use of second derivative (Hessian) matrix makes the method the most robust optimization method avoiding local optima
- Relatively few iterations for convergence

The method is ideally suited for neural network training with orthogonal input variables, e.g., with principal component scores.

Application of optimization algorithms to neural network training for power plant monitoring is described before [4]

Kalman Filtering Algorithm

The supervised training of a multi-layer perceptron is an adaptive nonlinear optimization problem for system identification. The function subject to minimization is the function of error that is the measure of deviations of the network outputs from desired or targets outputs. It is expressed as a function of the weight vector **w** representing the synaptic weights and thresholds of the network. The standard back-propagation algorithm is a basic optimization algorithm. With its simple form its convergence properties are relatively poor concerning speed and trappings at the local minima. To improve its convergence properties alternative approaches are implemented borrowing techniques from advanced adaptive non-linear least mean square estimation. These techniques are collectively referred to as *recursive least-squares* (**RLS**) algorithms that are the special case of the Kalman filter. Since the Kalman filter equations are for linear systems, in the neural network training this linear form should be modified. This modified form is known as *extended Kalman filter* and briefly described below.

Kalman filtering can be used for adaptive parameter estimation as well as optimal state estimation for a linear dynamic system. In the application of Kalman filtering to neural network training the parameters are the network weights which are considered as states and the estimation of the states is performed using the information applied to the network. However this information cannot be used directly as the size of the covariance matrix is equal to the square of the number of weights involved. To circumvent the problem we partition the global problem into a number of sub-problems where each is at the level of a single neuron. Then, with the exclusion of the sigmoidal non-linearity part, the model is linear with respect to synaptic weights. Now, we consider neuron i, which may be located anywhere in the network. During the training, the behavior of neuron i represent a nonlinear dynamical system so that the appropriate state and measurement equations are

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n)$$

$$d_i(n) = \varphi(\mathbf{y}_i^T(n)\,\mathbf{w}_i(n)) + z_i(n)$$

where the iteration n corresponds to the presentation of the n- th patter; $\mathbf{y}_i(n)$ is the input vector of neuron i; $\varphi(x)$ is the sigmoidal non-linearity. In the Kalman filtering $z_i(n)$ is the measurement noise which is assumed to be Gaussian white for the optimality. The weight vector $\mathbf{w}_i$ of the optimum model for the neuron i is to be estimated through training with patterns.

For linearization, the measurement equation is expanded to Taylor series about the current estimate $\mathbf{w}_i(n)$ and linearized as

$$\varphi[\mathbf{y}_i^T(n)n\ \mathbf{w}_i(n)] \approx \mathbf{q}_i^T(n)\,\mathbf{w}_i(n) + \{\varphi[\mathbf{y}_i^T(n)\,\mathbf{w}_i(n) - \mathbf{q}_i^T(n)\,\mathbf{w}_i(n)\}$$

where

$$\mathbf{q}(n) = \frac{\partial\varphi[\mathbf{y}_i^T(n)\,\mathbf{w}_i(n)]}{\partial\mathbf{w}_i(n)}\Bigg|_{\mathbf{w}_i(n)=\mathbf{w}_i(n)}$$

that results in for a sigmoid function

$$\mathbf{q}(n) = o_i(n)\,[1- o_i(n)]\,\mathbf{y}(n)$$

where $o_i(n)$ is the output of neuron i corresponding to the weight estimate $w_i(n)$; y(n) is the input vector. In the Taylor expansion the first term at the right hand side represents the linearization and the second term is the modeling error. Neglecting the modeling error we arrive at

$$d_i(n) = \mathbf{q_i}^T(n)\, \mathbf{w}_i(n) + z_i(n)$$

In the RLS algorithm, the measurement error $z_i(n)$ is a localized error. The instantaneous estimation of it is assumed to be

$$z_i(n) = -\frac{\partial E(n)}{\partial o_i(n)}$$

where E(n) is the error function defined by

$$E(n) = \frac{1}{2} \sum_{j \in O} [d_j - o_j]^2$$

where O represents the neural network output space. The resulting solution is defined by the system of recursive equations of the standard RLS algorithm that are

$$\mathbf{Q}_i(n) = \lambda^{-1}\, \mathbf{P}_i(n)\, \mathbf{q}(n)$$

$$\mathbf{k}_i(n) = \frac{\mathbf{Q}_i(n)}{1 + \mathbf{Q_i}^T(n)\, \mathbf{q}(n)}$$

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + z_i(n)\, \mathbf{k}_i(n)$$

$$\mathbf{P}_i(n+1) = \lambda^{-1}\, \mathbf{P}(n) - \mathbf{k}(n)\, \mathbf{Q_i}^T(n)$$

where n is the index parameter of iteration, each iteration being the introduction of one individual pattern from an epoch. $\mathbf{P_i}(n)$ is the current estimate of the inverse of the covariance matrix of q(n), and k(n) is the *Kalman gain*. The parameter λ is a *forgetting factor* less but closer to unity. The very last equation above is called *Riccati difference equation*. With the above recursive algorithm each neuron in the network receives its own effective input $\mathbf{q}(n)$ thereby possesses its own covariance matrix $\mathbf{P_i}(n)$.

In the actual Kalman algorithm in contrast with the exponential weighting of all estimation errors, a process noise is introduced in the state equation as

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \mathbf{v}_i(n)$$

where $\mathbf{v_i}(n)$ is the process noise which is white and assumed to be Gaussian for optimality. In this case the solution of the system equations presents a solution similar to those from RLS estimation as

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \mathbf{K}_i(n)\,[\mathbf{d}_i(n) - \mathbf{q}_i^T(n)\,\mathbf{w}_i(n)]$$

$$\mathbf{Q}_i(n) = \mathbf{P}_i(n-1) + \mathbf{R}(k-1)$$

$$\mathbf{K}_i(n) = \frac{\mathbf{Q}_i(n)\,\mathbf{q}(n)}{\mathbf{Z}(k) + \mathbf{q}(n)^T\,\mathbf{Q}_i^T(n)\,\mathbf{q}(n)}$$

$$\mathbf{P}_i(n) = \mathbf{Q}_i(n) - \mathbf{K}_i(n)\,\mathbf{Q}_i(n)$$

where **R** and **Z** are the auto-covariance matrices of the process noise and measurement noise, respectively; $\mathbf{K}_i(n)$ is the Kalman gain; $\mathbf{P}_i(n)$, is the error covariance matrix indicating the variances of the estimated parameters. The difference $d_i(n) - \mathbf{q}_i^T(n)\,\mathbf{w}_i(n)$ is called innovation and is easily computed for the output neurons. However for the hidden layer - $\partial E(n)/\partial o_i(n)$ can be used to replace the difference appearing in the state equation during the iterative period.

Kalman algorithm is relatively superior to other type of learning algorithms since it is a model-based algorithm and the modeling errors are counted in the measurement errors. On the other hand, the learning process during the neural network training is a stochastic process and this will be shown shortly afterwards. The process noise in the model considers the stochastic nature of the learning process and thereby it plays important constructive role on the convergence properties of the adaptive process. Due to these features of Kalman filtering, the convergence is much faster and undesirable excessive learning called memorizing incidents in neural network training is avoided as the memorizing badly deteriorates the generalization capability of the network. Additionally, the fast learning capability and ability of modeling stochastic signals make the Kalman filtering algorithm essential learning algorithm for using the neural networks for stochastic signal processing. Application of Kalman filtering algorithm to neural network training for power plant monitoring is described before [5,6].

To see the stochastic nature of the learning process consider a vector x of independent variables and a scalar d which is independent variable. If we have N measurements of **x**, we have $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_N$ and correspondingly $\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_N$. The neural network model is expressed by

$$\mathbf{d} = g(\mathbf{x}) + \mathbf{e}$$

where $g(\mathbf{x})$ is some function of the vector **x** and **e** is random expectation error. In this model, taking the expectation of both sides, the function g(x) is defined by

$$g(\mathbf{x}) = E[\mathbf{d}|\mathbf{x}]$$

The conditional expectation defines the value of d that will be realized on the average given a particular realization of **x**. Let the actual response of the network is defined by

$$y = F(\mathbf{x},\mathbf{w})$$

where w represents the synaptic vector. The synaptic weight minimization for the cost function $J(\mathbf{w})$ used in neural network training

$$J(\mathbf{w}) = E[(\mathbf{d} - F(\mathbf{x},\mathbf{w}))^2]$$

would also minimize the multiple integral

$$E[(g(\mathbf{x}) - F(\mathbf{x},\mathbf{w}))^2] = \int f(\mathbf{x}) [g(\mathbf{x}) - F(\mathbf{x},\mathbf{w})]^2 d\mathbf{x}$$

where $f(\mathbf{x})$ is the probability density of $\mathbf{x}$ and note that, the function subject to minimization is a random variable. This result therefore, indicates the statistical nature of the learning process even though the different patterns of a particular realization of $\mathbf{x}$ are deterministic. From the probability theory viewpoint, the difference between the input-output pairs of $\mathbf{x}$ used for the training obeys an arbitrary distribution $f(\mathbf{x})$ and therefore do not take a functional relationship between input and output deterministically. $f(\mathbf{x}|\mathbf{w})$ being the density function parameterized by $\mathbf{w}$, the model $f(\mathbf{x}|\mathbf{w})$ with good generalization capability should approximate the true distribution $f(\mathbf{x})$. In other words, the learning is dependent on the probability density of the input vector where the network performance should not depend on the network structure, i.e., $f(\mathbf{x}|\mathbf{w})=f(\mathbf{x})$. It should only be dependent on the statistical properties of the input vector itself, if the appropriate network model is used for training.

2.2.2. Unsupervised Learning Algorithms

Self Organizing Feature Map Network

Feature mapping converts patterns of arbitrary dimensionality into the responses of one or two-dimensional arrays of neurons called *feature space*. A network that performs such a mapping and the related algorithm are referred to as feature map and self-organizing feature mapping (**SOFM**), respectively. These networks are based on competitive learning where the output neurons of the network compete among themselves to be activated so that only one output neuron is on at any one time. The output neurons that win the competition are called winner-take-all neurons. In addition to achieving dimensionality reduction, also topology-preserving map that represents the neighborhood relations of the input patterns is obtained. That is, the similar patterns in some sense should give outputs topologically close to each other. The output layer in this map commonly one or two-dimensional virtual array in the form of a lattice. Different input features are created over the lattice. Each component of the input vector $\mathbf{x}$ is connected to each of the lattice nodes as shown in Fig.3. To describe the SOFM algorithm the input vector $\mathbf{x} = [x_1, x_2, ..., x_p]^T$ and the synaptic weight vector of neuron j , $\mathbf{w} = [w_{j1}, w_{j2}, ..., w_{jp}]^T$, j=1,2,..., N , is considered. n is the total number of neurons. To find the best match of the input vector $\mathbf{x}$ with the synaptic weight vectors $\mathbf{w_j}$, we compare the inner products $w_j^T x$ for j=1,2,...,N and get the largest. In the formulation of an adaptive algorithm, it is convenient to normalize the weight vector $\mathbf{w}_j$ to constant Euclidean norm. In this case the best matching criterion is equivalent to the minimum Euclidean distance between vectors.

One approach to training a self-organizing feature map network is to use the Khonen learning rule with updated weights going to the neighbors of the winning neuron as well as to the winning neuron itself. The topological neighborhood of the winning neuron i is shown in Fig. 5 and it is represented as Λ_i (n) indicating its discrete time (iteration) dependence.

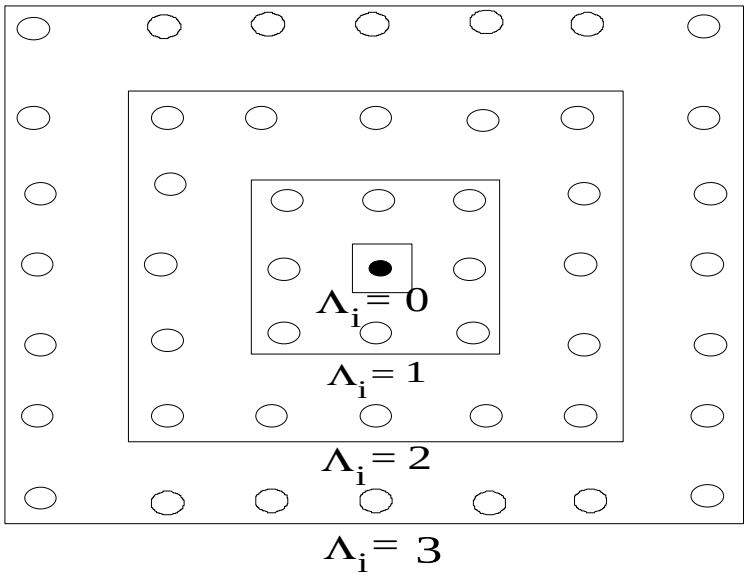

Fig.5: Square topological neighborhood Λ around *winning* neuron i at the middle

The basic self-organizing learning algorithm can be summarized as follows.

1. *Initialization*. Choose random values for the initial weight vectors $\mathbf{w_j}(0)$

2. *Select Pattern*. Draw a pattern $\mathbf{x}$ from the input patterns formed by a set of sensory signals

3. *Similarity Matching*. Find the best matching, that is, winning neuron $i(\mathbf{x})$ at time n, using the minimum-distance Euclidean criterion:

$$i(\mathbf{x}) = \arg_j \min \| \mathbf{x}(n) - \mathbf{w}_j \| \quad , \quad j=1,2,...,N$$

where N is the number of neurons in the two-dimensional lattice.

4. *Updating*. Adjust the synaptic weight vectors of all neurons, using the update formula

$$\mathbf{w_j}(n+1) = \begin{cases} \mathbf{w_j}(n) + \eta(n)\,[\,\mathbf{x}(n) - \mathbf{w_j}(n)\,] & j \in \Lambda_i(n) \\ \mathbf{w_j}(n) & \text{otherwise} \end{cases}$$

for the winning neuron where $\eta(n)$ is the learning-rate parameter; $\Lambda(n)$ is the neighborhood function centered around the winning neuron $i(\mathbf{x})$ and it is illustrated in Fig.5.

5. *Continuation*. Return to the step 2 and continue until no noticeable changes in the feature map are observed.

With this learning rule, nearby nodes receives similar updates and thus end up responding to nearby input patterns. The neighborhood function and the learning constant η are changed dynamically during learning. As result of the of weight adjustment, a planar neuron map is obtained with weights coding the stationary probability density function p(**x**) of the pattern vectors used for training. The ordered mapping of patterns establishes the weights indicated $\mathbf{w}_{ij}$. The self-organizing learning process is stochastic in nature so that the accuracy of the map depends on the number of iterations of the SOFM algorithm. Also, the learning parameter η and the neighborhood function Λ_i are critical parameters. That is their appropriate selection is essential for satisfactory performance after the training.

2.3. Sensitivity

Sensitivity analysis investigates the effect of changes in input variables on output predictions. It is based on the perturbation of a system parameter at a point of interest in the form of partial derivatives $\partial y_j/\partial x_i$ where y_j represents the j- th network output and x_i represents i-th network input where y is expressed by

$$y_j = f(x_1, x_2, ..., x_n)$$

Also one can define dimensionless sensitivity coefficient (dy/y)/(dx/x) which represents ratio of the percentage changes in the parameters. Sensitivity is a measure of contribution of an input to let the network recognize a certain pattern in a non-temporal neural network application where the input-output mapping is static. That is both the input vector **x** and the output vector **y** are represent spatial pair of patterns that are independent of time as this is the case with the benchmark data. Ideally, the relative sensitivities of each network's input-output pair are desired to be about the same for the robustness of the network. A robust network is one that estimates the correct output for a respective input that contains an error or missing data, without degrading the output estimates of the other parameters at the network's output. To achieve robustness, certain degree of noise is superimposed on training data set. In this case the training set gains some probabilistic features subject to estimation. Referring to this, a commonly used probabilistic sensitivity measure that relates to input and response probability distributions in terms of the standard deviations is ($d\ \sigma_y/d\sigma_x$). The importance of this due to the information it provides on the degree of response uncertainty or variability reduction as a function of reduction of the input uncertainty. Similarly, a probabilistic measure associated with a change in the mean value is useful in examining the effect of the uncertainty in the mean value of an input random variable. Probabilistic sensitivity measures identify essential input parameters that contribute to the uncertainty in the estimation of the network. The relative importance or ranking of input random parameters can be used in making decisions leading to uncertainty reduction.

3. ANALYSIS OF THE RESULTS REPORTED

The benchmark analysis results are globally presented at SMORN VI, in the form of an extended summary where also the analysis results of the benchmark organizers are illustrated. The full documentation of the related Extended Summary Report is given in the appendix A and the report has also been documented in a recent OECD-NEA publication [7]. In this section the analysis of the results reported by the participants are presented. The presentation comprises a concise description of the documented results sent by the participants pointing out also the key

features. For the detailed information on any of these descriptions reference is made to the full documentation which is included in the Appendix B. Below, is the concise evaluation of the submitted reports where the presentations are in alphabetic order referring to the respective countries.

1. H. Mayer, P. Jax and U. Kunze (Germany)

The participants used an algorithm coded in C $_{++}$ and developed by them. The extensive analyses have been performed and their results are reported. For training the network standard back-propagation algorithm for multi-layer perceptrons (MLP) networks is used. For the prescribed network structures different back-propagation learning strategies i.e., variation the learning rate η and momentum rate μ, are employed. From the benchmark analysis submitted, the following observations are made

- Various activation-functions for neurons are employed. These are, Sigmoid Fermi Function, Limited Sine Function, Unipolar Ramp Function, Gaussian Function.

- Standard back-propagation algorithm with some learning variants is employed. These are mainly
  - Super SAB
  - Delta-bar-delta
  - Resilient Back-propagation

- In addition to the standard back-propagation algorithm, some variants of this algorithm are also exercised where insufficient or no convergence was obtained. Among these are
  - Zero-point search method
  - Batch learning in constant η
  - Learning with adapted global η
  - AMBP learning
  - Quickprop (back-propagation) learning
- The highest convergence rate and best generalization are achieved with the inputs when
  - Error function is based on L_2-norm
  - Each active neuron receives a constant excitation that is also subject to weighting, i.e., threshold
  - Excitation is achieved in on-line learning by stochastic presentation of each pattern with an epoch
- Input and output signal normalizations are performed between 0 and 1
- Errors are expressed as deviations
- Basic sensitivity analysis is considered

The extensive study results are reported and they indicated that the participant have endeavored to have thorough understanding of back-propagation neural network performance by using the actual plant data and they carried out researches beyond the formal benchmark analysis tasks; repeating the analyses with all available signals, for example. Their results are among the outcomes of high performance category, considering the reported benchmark results of participants in perspective. The standard back-propagation (BP) learning algorithm is rather popular because of its simplicity. However it has rather poor final convergence properties, although the initial convergence properties are satisfactory, and therefore it needs a lot of elaboration during training. The elaboration is carried out and therefore extensive analysis results are obtained. The deviations of the estimated signals by the network from the actual values are relatively small and generalization capability is quite satisfactory. However, some BP learning strategy might still improve them. This learning strategy should still be a dynamic

one as far as the learning parameters are considered in a BP approach. Although the mathematical treatment is somewhat involved, basically, due to the learning process is a stochastic process, the stochastic presentation of the training examples should be randomized (shuffled) from one epoch to other for better performance of BP algorithm. This general result is also verified experimentally in the reported research. Although several variants of the BP algorithm are investigated, any recommended reference to one of them among the others is not made for eventual condition monitoring applications.

2. U. Fiedler (Germany)

The participant used a commercially available NeuralWorks II/Professional Plus algorithm by NeuralWare, Inc. From the benchmark analysis report submitted, the following observations are made

- Normalization is performed and bipolar inputs are used
- Standard back-propagation (BP) algorithm is used with sigmoid activation function. Momentum term is included into the learning algorithm although momentum coefficient is relatively low and learning parameter is high. Learning parameter is decreased in the advanced stage of learning.
- Errors are reported as root mean square (RMS) and MAX-error in % in contrast with deviations from the nominal value.

From the inspection of the results reported it is to conclude that, the additional elaboration on the training strategies would be suggested to improve the reported results.

3. Y. Ding (Germany)

The participant used a self-developed algorithm in Window environment. From the benchmark analysis report submitted, the following observations are made

- The neural network in use is especially designed for false alarm reduction in a monitoring task. However it is modified for the present benchmark.
- Normalization is made between 0 and 1 but 10% over ranging
- The order in which the examples are presented to the network is randomized
- Relatively high learning rates are used

The results reported indicated the global convergence ability of the algorithm used. However, the deviations between the network estimates and the nominal input counterparts at some parts of the learning period are relatively high and this can be improved by carrying out some elaborated learning strategies. Especially, the modification of the learning parameters, i.e., learning rate η and the momentum coefficient α during the learning process should be varied for enhanced training. Increased training patterns gave better results for the training ranges as well as the ranges beyond the prescribed training ranges. In spite of overall satisfactory performance of the network used, a better performance requirement for this network in critical applications is mainly attributed to the original design of the network that is tuned for the benchmark analysis. Additional elaboration on the training strategies would apparently improve the reported results for a desirable level of performance.

4. S. Thangasamy (India)

The participants used an algorithm in C language in UNIX environment. From the benchmark analysis report submitted, the following observations are made

- The algorithm makes use of a sigmoid function.
- Batch mode of training is implemented. In the batch mode of learning, weight updating is performed after the presentation of all training examples that constitutes an epoch.
- The cost function which is conventionally sum of the squared errors divided by the number of patterns is further divided by the number of neurons to obtain an average square error per neuron. The latter is used to determine the success of training and terminate the process of optimization over the weight space.

- Maximum number of iterations is also used as a parameter to help intervene during optimization in situations of very low error thresholds or the nature of the problem leading to poor convergence.

The results reported indicated the global convergence ability of the algorithm used. However, the deviations between the network estimates and the nominal input counterparts at some parts of the learning period are relatively high. Increased training patterns gave better results for the training ranges as well as the ranges beyond the prescribed training ranges. In spite of overall satisfactory performance of the network used, a better performance requirement for this network in critical applications is mainly attributed to the normalization of the data, presentation of the patterns to the network and the static nature of the learning parameters η and α. Additional elaboration on the training strategies would apparently improve the reported results.

5. G.S. Srinivasan and Om Pal Singh (India)

The participants used back-propagation algorithm. Among the basic benchmark tasks, multi-input single-output structure was subject to study and the relevant results are reported. From the benchmark analysis report submitted, the following observations are made
- Sigmoidal activation function was assigned at the neural nodes
- The initial weight values were randomly chosen from sampled background noise of steam generator environment of a fast reactor. The weight values were normalized between -1 and +1.
- Momentum term in learning algorithm was not employed and still fast convergence is reported.
The results reported indicated the global convergence ability of the algorithm used. However, deviations between the network estimates and the nominal input counterparts at some parts of the learning period and for some certain signals are relatively high and vice versa. This is attributed to the initial weight selection strategy and the missing momentum term in learning. The function of the momentum term during learning is twofold. One is that it accelerates descent in steady downhill directions. The second is that it prevents the learning process from terminating in a shallow local minimum on the error surface in the multidimensional space. The initialization of the synaptic weights and threshold levels of the network should be uniformly distributed inside a small range. This is because to reduce the likelihood of the neurons in the network saturating and producing small error gradients. However, the range should not be made very too small, as it can cause the error gradients to be very small and the learning therefore to be initially very slow. Additional elaboration on the training strategies would apparently improve the reported results.

6. M. Marseguerra and E. Padovani (Italy)

The participants have submitted a detailed analysis report of the benchmark tasks they executed. They used an algorithm of their own coding. The neural network (NN) training performed with the back-propagation (BP)-algorithm as follows.

- The initial values of the connection weights w_{ij} are randomly chosen in the interval -0.3, +0.3.
- Batch training mode is employed where the w_{ij} are updated after each batch of 10 training patterns.
- Each batch of training patterns is repeatedly (15 times) presented to the network.
- The whole training lasts N batches, each repeated 15 times. Thus the network receives a total of 10x15xN training patterns and its weights are updated 15xN times. A typical figure for N is selected to be $N=1.6\ 10^5$.
- After the 15 repetitions of each batch (and therefore after 15 updating of the weights) the learning and momentum coefficients are varied according to a sigmoid function from the initial values α_o and η_o to zero.
- The initial values α_o and η_o relating to the neural network for generated electric power (GEP) estimation are 0.7 and 0.3, respectively. Those for the auto-associative NN are 0.5 and 0.6, respectively. The above values have been selected over a two-dimensional grid of points within the unit-square.

From the benchmark analysis report submitted, the following observations are made.

- The cost (error) function subject to minimization is the square of the difference between the desired and the computed (estimated by NN) output, averaged over the training set, i.e., number of patterns in one epoch.
- Normalization of the signals is performed between 0.2 and 0.8.
- Errors between the estimated values and their counterparts are expressed in terms of deviations.
- Sensitivity analysis is performed in great detail and a substantial discussion is presented. They are reported as a set of the partial derivatives of outputs with respect to inputs.

Since neural network is a non-linear device, the sensitivities, besides depending on the synaptic weights that are constant once the NN is trained, also depend on the particular input pattern. Therefore, it should be expressed as an average value considering all input-output pattern pairs forming an epoch. Generally, a data set involving a set of input-output pairs and being used a feed-forward neural network training does not contain all the information about the dynamic system where the set comes from. Therefore the sensitivities between input and output represent individual neural network attribution for the signals in use rather than those representing the plant-sensitivity. For robust training these attributions are approximately expected evenly distributed and independent of the initial point used to start training. The verifications on the sensitivity considerations are successfully carried out in this benchmark report with fruitful relevant discussions. The results reported indicated that the participant have endeavored to have thorough understanding of back-propagation neural network performance by using the actual plant data and they carried out researches beyond the basic benchmark analysis tasks and extended to optional tasks. Their results are among the outcomes of high performance category, considering the reported benchmark results of participants in perspective. The deviations of the estimated data by the network from their actual counterparts are relatively small and generalization capability is quite satisfactory. However, by some back-propagation learning strategy they can still be improved. This learning strategy should still be a dynamic one as far as the learning parameters are considered in a BP approach.

7. K. Nabeshima (Japan)

The participant has submitted his benchmark analysis results, all the prescribed benchmark tasks having been performed. Additionally, he gave his results on the sensitivity analysis. He used a code originally developed for on-line plant monitoring where the network gradually changes its characteristics by updating the weights. Standard BP algorithm with sigmoidal non-linearity and momentum term was employed in his analysis where both input and output signals are normalized to be in between -0.5 and +0.5 for inputs and 0.1 and 0.9 for the output signals. He reported the weight values and the learning errors of the trained networks used for the benchmark tasks together with the minimum and maximum estimated signal values for each case. He repeated the similar experiments not only for the prescribed network structures for the benchmark but also using various number of hidden layer nodes as optional investigation for finding optimal number of hidden layer nodes in each case, starting from five. He reported both, the global training error and the corresponding error in the test (recall) phase outcomes. This was carried out for both, auto-associative and hetero-associative cases aiming to obtain a better network performance. Training errors decreased with the increasing number of hidden layer nodes although this was not the case for the recall-phase outcomes that it makes the case inconclusive. He eventually concluded that the number of hidden layer nodes is not a sensitive parameter for learning. However, since the learning error is not the only factor for the determination of the performance, further elaboration on this issue is needed.

He carried out sensitivity analysis using the ratio of finite variations of the signals in place of analytical derivatives. The sensitivity results are rather uneven indicating some need of consideration on this issue as the even signal contributions are desirable for a better network performance. Because of gross computation of the derivatives, some additional errors might have affected the results.

Following the training, the reported errors are rather small so that, his detailed reported results are expectedly quite satisfactory while the results are among the outcomes of high performance category, considering the reported benchmark results of participants in perspective.

8. Dai-Il Kim (Republic of Korea)

The back-propagation algorithm used in this participation is coded in C and run in a Pentium-90 computer. The activation function considered is a bipolar function with a constant slope. The slope is 1.0 for output neurons and 0.8 for the hidden neurons.
From the benchmark analysis report submitted, the following observations are made.
- Normalization of the signals was performed between -0.5 and +0.5.
-Constant learning rate is employed where η=0.068
- Errors between the estimated values and their counterparts are expressed in terms of deviations.
- No momentum term is employed.
- Some optional tests are performed for the robustness verification of the network and the sensitivity analysis is employed for this process.
The results reported indicated the global convergence ability of the algorithm used. However, the deviations between the network estimates and the nominal input counterparts at some parts of the learning period and for certain signals are relatively high and vice versa. This is attributed to the constant learning rate and the missing momentum term in learning. The function of the momentum term during learning is twofold as was explained before. Robustness of a neural network can be defined as the stability of the learning process being independent of the starting point of the learning process in the multidimensional parameter space. Additionally,

the sensitivities are desirably expected approximately evenly distributed. In this contribution the robustness verification is performed with the help of sensitivity analysis. To this end, the investigations are extended to the network of 12:14:12 structure apart from the nominal auto-associative network of 12:8:12 structure prescribed in benchmark task execution. The sensitivities reported are rather uneven and rather low. Additional conventional elaboration on the training strategies would apparently improve the reported results.

9. E. Barakova (The Netherlands)

The standard back-propagation algorithm is used in this participation. For this, commercial software called 'InterAct' is employed. Because of the limitations of the available software, the benchmark data was considered to be a temporal data and accordingly very limited analysis was performed with temporal processing. However the benchmark data at hand is a combination of different operations at different times and the data sets are the in the form of patterns to apply to the neural network as spatial input patterns rather than temporal sequences. Therefore the deviations between the estimated signal values and the given counterparts are reported to be relatively high, as one should expect.

Temporal processing requires that the neural network should have dynamic properties that makes it representative to time-varying signals. For this purpose, time delays are introduced into the synaptic structure of the network and to adjust their values during the learning phase. Since the benchmark tasks concern only spatial learning process in contrast with temporal (time-dependent) learning process the reported results are to be considered and re-reported accordingly.

4. FURTHER STUDIES WITH THE BENCHMARK DATA

In this section some further analyses carried out with the benchmark data will be presented. These include advanced learning algorithms and the sensitivity analysis and they are given in the following subsections.

4.1. Advanced Learning Algorithms

As it is noted from the reported benchmark results, all participants used the standard back-propagation (BP) algorithm as a basic training algorithm. However, as it was pointed out earlier, this should be replaced with more advanced training methods to enhance the training. In corroboration with these statements, it is interesting to note that although the training algorithm is the same and basic one, there is rather big performance difference among the reported results. This is because of the selection of the parameters and necessity of their modification during training while a systematic approach for such a treatment cannot be prescribed in advance as this is dependent on the data and the circumstances. The essential algorithms for this purpose are described in some detail in the preceding section. With respect to this, some additional studies have been carried out. In this context, some direct results together with the associated results obtained with the implementation of these algorithms are presented below.

The benchmark data analysis results obtained by BP algorithm were presented earlier and they are illustrated in the extended summary report [8]. In this study, advanced nonlinear optimization methods for training together with the enhanced sensitivity analyses are considered. Additionally, to emphasize the generalization capability of these algorithms the

reported analyses here are carried out also slightly different than those carried out for benchmark standard benchmark tasks. In particular, in the benchmark task for the auto-associative neural network with multiple signal data, the training is carried out for initial 600 patterns. The associated recall was performed for the total number of patterns of 1169 in contrast with 942 patterns used in the standard benchmark task (in place of 1169 patterns in Task **a-1**).

To begin with, the results obtained with the BP algorithm are given in Fig.6 where the learning is carried out first 1000 patterns and the task is designed to be **BB-1**. In this analysis, a set of uncorrelated random data, i.e., noise, is imposed on the benchmark data. The effect of this noise introduction on the data can be explained in the following way. Since the benchmark data is obtained from actual power plant, it contains measurement errors. During the network training this measurement errors affect the training and degrade the learning performance. In the extreme case these errors are also learned where this situation is termed as *memorizing*. To avoid this, the learning process should be limited in the case of working with measurement data. One method for this limitation is the random data introduction to the data. As result of this, the measurement errors are not traced in the learning process of the network and the net result is some reduction of the estimation errors by the network. This result is seen in Fig.6, where the standard deviation of the noise introduced on a signal is equal to the 5% of the maximum value of the respective signal among the total number of patterns and the standard deviation is kept constant for that signal throughout the patterns. Attention is drawn that, by the noise addition the estimation errors by the network are slightly reduced, for the cost of missing some minor details also available in the given data. In connection with this, reference is particularly made to one of the peculiarities present in the benchmark data, the peculiarity being the rinsing operation. This operation is generally done for thermal performance enhancement and this is included into the benchmark data twice consecutively but in opposite phases as this is seen in patterns around 800. This particular operation belongs to the normal operation while the operational conditions are slightly different than those involved during routine operation. The trained neural network without the rinsing-process experience is expected not to follow this unknown operation. However, in this case the difference between the estimated signal value and the signal itself would be relatively large so that this particular operation is detected. This is clearly seen in the illustrations of the benchmark analyses. Referring to this detail, the noise introduction to the data partly obscures the identification of this detail. However, this consequence can be assessed positively since the rinsing process belongs to normal operation and the estimation error should not be big as that could lead to reporting false alarms. The trade-off between these two considerations should be kept in view. In the framework of the present study, the other implications of the noise introduction on data will be illustrated shortly afterwards in connection with *sensitivity*. A similar test as above but taking 600 patterns for learning is given in Fig.7 where the task is designed to be **BB-2**. To show the signals and the superimposed noise magnitude involved in scale together, the same analysis above is illustrated again in Fig.8 where the task is designed to be **b-1**. In Fig.8 particularly, in place of the estimated patterns, the patterns introduced at the neural network input are presented. The errors seen in the figure are irrelevant but they are there because of the plotting routine bay the algorithm used for plotting and therefore the errors are to be out of concern and subject to omission in this particular case. However, here a point noteworthy to mention is the peculiarity of the operation represented by the patterns after 942. This operation is termed as *stretch-out* and briefly described blow in connection with the further analyses. Referring to the stretch-out operation, during the learning process using the patterns up to 1000, this particular operation is explicitly included into the training. The resulting corresponding estimation i.e., estimation beyond the pattern 942, by the network is much improved relative to those obtained using the patterns until 942 from the beginning, for learning. The implications of this difference is presented below together with the further analyses related.

The analyses are carried out using the advanced training algorithms as these algorithms are described in the preceding section. These are namely, nonlinear optimization algorithms, Kalman filtering algorithm and radial-basis-functions (RBF) network algorithm (self-organizing and orthogonal-least-squares). In this context, some illustrations of auto-associative benchmark data analysis results with variable-metric nonlinear optimization algorithm are shown in Fig.9 where the task is designed to be **C-2.** The range of training spans first 600 patterns of the data set and recall is carried out over the total number of patterns, i.e., 1169. The back-propagation counterparts of this analysis are reproduced here for convenience in Figs.10 and 11 while they are presented in the extended summary report [7] earlier (**Benchmark Tasks a-1 and a-2**). The counterpart benchmark tasks are in the form of two analyses in place of one due to the range of the analyses; namely, in the **Task a-1**, the learning spans first 600 patterns and the corresponding recall is carried out over 942 patters from the beginning. In the **Task a-2**, the learning spans first 1000 patterns and the corresponding recall is carried out over the total 1169 patterns. In addition to the first peculiarity of the data described above, here the attention is especially drawn to the second peculiarity which concerns a particular plant operation in the form of extended plant operation which is referred to as 'stretch-out'. This type of operation is quite peculiar to the PWR plant from which the benchmark data prepared. This operation is performed towards the end of the fuel-cycle to extend the plant operation for some time. Referring to the first principle of the Thermodynamics, this is accomplished by keeping the temperature difference between the hot-leg and cold-leg about the nominal range although the respective temperatures themselves are below their nominal ranges. In this respect, the operation belongs to the category of normal operation and the major differences between the operation in the course of fuel cycle and at the end of the cycle are the temperatures as far as the neural network is concerned. Because of this very reason, in the benchmark, in place of individual hot-leg and cold-leg temperatures, the differences (two differences because of two primary loops) are considered for training and thereby the major difference between the operations have been eliminated. The basic implication of this is that the total set of benchmark data being considered is expected to be appropriate representative of one complete fuel-cycle power plant operation, the data being made exclusively suitable for training for the purpose of plant monitoring by neural networks. From the inspection of Fig.9, the enhancement in the pattern estimation and the verification of the expected generalization capability achieved by using an advanced training method is rather outstanding.

In addition to the comprehensive analyses performed by the advanced training algorithms some illustrations of which described above, further analyses are carried out using the same benchmark data and algorithms above with the superimposition of the uncorrelated random data that might loosely be termed as *noise*. Because of the popularity of BP algorithm which is uniquely used by the participants, a comparison between BP and the advanced algorithms would be of great concern and very informative. This illustrations are presented in Fig.12 where the task is designed to be **C-3** and Fig.13 where the task is designed to be **C-1.** In Fig.12, the advanced training algorithm considered is the quasi-Newton optimization algorithm. The learning part uses 600 patterns from the beginning and recall phase is carried out over the total number of patterns.. The comparison reveals that, on one hand, the noise introduction let the estimation errors diminish since by doing so, the effects of the measurement errors are minimized as this was explained above. On the other hand, with the standard deviation of the noise introduced on a signal that is equal to 3% of the maximum value of the respective signal of the pattern, the generalization ability of the network trained by the advanced training algorithm is slightly inferior during the stretch-out operation. The generalization ability of a neural network can explicitly be tested during the stretch-out operation as this was explained before. The exercised deterioration of the generalization ability in this case is a result that one should normally expect. This is due to the increased uncertainty imposed on the data.

Naturally, while the uncertainty due to noise, to some extend, eliminating the measurement error effects on the network for nominal operational conditions, it introduces also increased uncertainty to the network for the extrapolated operational conditions, like the *stretch-out*, in this case. Fig.13 represents the similar studies carried of with the standard back-propagation algorithm. Comparison of Fig.12 with Fig.13 indicates the favorable estimations with the advanced training methods relative to those obtained by the BP algorithm.

For easy inspection of the above described studies, the plots of twelve estimated auto-associate network outputs with and without noise imposition are collectively given by Figs.14-37. Here, the learning part comprises 600 patterns from the beginning of the benchmark data and the learning algorithm is based on advanced nonlinear optimization by quasi-Newton. For the comparison of the noise magnitude with the magnitudes of the pattern components throughout the total number of patterns, they are presented together and illustrated in Figs.38-49.

It is noteworthy to mention that, noise impositions are made individually on each signal. This implies that, the temperature differences used during the benchmark task executions are carried out using these in-advance noise imposed individual temperature signals. Therefore the resultant noise level of these signals are relatively high as this can be noted in Fig.8 and also, explicitly in Figs.41 and 42. However, it is interesting enough that, the corresponding estimations by the auto-associative network for these inputs are relatively better and satisfactory as these are to be seen in Figs.12, 13 or alternatively in Figs.29 and 30. This indicates that, following training, the decision boundaries in a multi-dimensional space of a feed-forward neural network should not necessarily be sharp for a better performance. In other words, fuzzy separation of boundaries may equally be functional in contrast with intuitive expectation. Noise imposed temperature signals subjected to form the difference in the benchmark are explicitly shown in Figs.50-53.

Enhanced neural network performances with the Kalman filtering and radial-basis-function (RBF) network training algorithms are obtained. They are similar to those obtained with the general nonlinear least-squares optimization methods with some subtle differences. Due to the approximate equivalence of these outcomes and also due to the volume of this report at hand, illustrations of these results are not here included. However, it is noteworthy to mention that Kalman filtering training algorithm is especially very fast relative to other advanced training methods mention made here already and this very algorithm inherently prevents the excessive training known as *memorizing* in the terminology of neural network training . Although the Kalman filtering is for linear systems and it is optimal in the least-squares sense for Gaussian signals, its adaptive parameter estimation form for neural network training is rather peculiar and superior, due to its inherent stochastic modeling properties. On the other hand Kalman filtering is the most suitable training algorithm for stochastic information processing by neural networks as it is used in the form of stochastic signal modeling, the model parameters being simply the neural network weights in this case [5,6]. The RBF network is an alternative to multi-layer perceptron (MPL) feed-forward neural network having the same multi-layer structure but better functional approximation properties due to peculiar activation functions (radial-base-functions) and the linearized output layer which provides computational simplicity in training. Due to the self-organized training stage, the training of RBF networks may need relatively more time for training, in this training mode among others. In return to this they can represent the given information more effectively due to clustering properties as this can play important role when a network is subject to training with high number of data representing information on different operational conditions.

4.2. Sensitivity Analysis

Sensitivity in the context of neural network training is defined and elaborated in the section II.3. In this section sensitivity analyses carried are presented. As the sensitivity is defined in the form of partial derivative the and averaged over the total number of patterns, such a computation is very close to computations in batch learning. Batch type learning is involved in especially general nonlinear optimization algorithms used for training where accumulated gradient information over the total number of patterns is used to determine the direction towards to minimum for convergence. Therefore it is rather convenient to carry out such analyses following the training accomplished by such algorithms. In the present study, the sensitivity analysis is carried out following the variable-metric optimization method for training. The relevant computations are documented below.

The first analysis results are given in Table I under the title *Sensitivity and Gradient Analysis* where the benchmark data is subjected to a detailed sensitivity analysis. In this analysis firstly, sensitivities are given in an order with respect to the input node sequence where the sensitivities are normalized with respect to the magnitude of the maximum sensitivity among the signals so that the maximum sensitivity value becomes unity. The same results are presented in an order with respect to their relative magnitudes. These two types of representations are presented in the same way for all signals used in the auto-associative network.

The sensitivity analysis results described in Table I, are followed by the gradients of the weights of the auto-associative network. With respect to each weight, all gradients are presented both, altogether and in detail. The gradient scheme given can be considered as a *mapping* of the learning and it gives important information about the training performance of the neural network. For the case for which the gradients are reported here, the magnitude of the gradients are rather low indicating the existence of a satisfactory minimum which is rather sharp in the multidimensional space with the positive indications of a rather satisfactory training performance.

The illustrations of the sensitivity analyses results in Table I, are presented in Figs. 54 and 55. The presentations are carried out in two ways; namely, Fig.54 describes the sensitivity with respect to each output node. The horizontal-axis is for the output nodes, and the vertical-axis is for representing relative contributions of each input signal to any output signal in a cumulative form that amounts to unity. The sensitivity pattern is clearly seen in this figure. This type of representation of the sensitivity is named in this study as **Sensitivity of Type I.** Fig.55, represents the sensitivity in a different form with additional information where the horizontal-axis is for the input nodes, and the vertical-axis is for representing relative contributions of each input signal to any output signal in a cumulative form which amounts to unity. This type of representation of the sensitivity is named in this study as **Sensitivity of Type II.** Rather desirable even distribution of the contributions is quite conspicuous.

The same analyses described above were repeated with the same data with noise superimposed on the data in the same way as described before. That is, the standard deviation of the noise introduced on a signal is equal to 3% of the maximum of that signal. For the total number of patterns the standard deviation is kept constant for that signal throughout the analyses. The results are presented in Table II in the same form as presented in Table I.

The sensitivity analysis results described in Table II are followed by the gradients of the weights of the auto-associative network. With respect to each weight, all gradients are presented. In this respect, the gradient scheme given can be considered as a *mapping* of the learning with noise, in this case and it gives important information about the training performance of the neural network. For the case for which the gradients are reported here, in contrast with the results

obtained without noise imposition, the magnitude of the gradients are relatively rather high indicating a gross minimum in the multidimensional space. However, even with this training, i.e., using noisy data, the performance of the network has only very slightly degraded as the performance of this network is already described above, in detail. The implication of this result is that, for a neural network to converge to a minimum defined by a cost (or energy) function is essential. However, it is possible to obtain a satisfactory performance from a neural network where the training process converges to a gross minimum. Incidentally, the extreme of converging to the absolute minimum may lead to an undesirable form of learning to some extend which is termed as *memorizing*, as mentioned before. The illustrations of the sensitivity analyses results with noisy data, given in Table II are presented in Figs. 56 and 57. The presentations are carried out in two ways for each case. Namely, Fig.56 describes the sensitivity with respect to each output node. The horizontal-axis is for the output nodes, and the vertical-axis is for representing relative contributions of each input signal to any output signal in a cumulative form which amounts to unity. This sensitivity representation is named as **Sensitivity of Type I**, above. The sensitivity pattern is clearly seen in this figure. Fig.57, represents the sensitivity in a different form with additional information where the horizontal-axis is for the input nodes, and the vertical-axis is for representing relative contributions of each input signal to any output signal in a cumulative form which amounts to unity. This sensitivity representation is named as **Sensitivity of Type II,** above. Rather even distribution of the contributions in both cases is quite conspicuous. Strictly speaking, from the comparison of Figs.56 and 57 with Figs.54 and 55, it is to conclude that with the addition of noise, the sensitivity of Type I is slightly improved. In return to this however, the sensitivity of Type II is slightly degraded. These results corroborate the previous analysis & evaluation results described previously above, in this work.

5. NEURAL NETWORK BENCHMARK FOLLOW-UP-96

The recommendations for a follow-up of the benchmark-95 will be emphasized in this section. This very view having been concluded already, in this study, some alternative proposals for the content and form of the follow-up are developed and they are presented point-wise, below. In these proposals the common feature is that all proposals are intended for neural network utilization primarily in nuclear industry to improve safety and operational cost-effectiveness. In parallel with this, all benchmark proposals endeavor to accomplish stochastic information processing as well as deterministic signals in the benchmark, although the latter is relatively easier to understand and implement. By means of this, the neural networks will be more of concern to reactor noise community and hence the implementations will also gain gravity in that direction so that the community would be provided with new potential tools and related possibilities. However, due to present early stage of the stochastic signal analysis by neural networks, the scope of such benchmarking would be relatively more carefully determined to encourage the participation. The detailed provisions of the specifications will be provided upon specific request.

5.1. Follow-up-96: NN for Condition Monitoring

This proposal concerns the preceding benchmark (Benchmark-95) subjected to study here. This is basically due to the difficulties that the benchmark participants, with very few exceptions, generally encountered during the task executions. Both, to encourage the new participation and to encourage the existing participants to improve and/or complete their work that they accomplished and submitted it is highly recommended continuing, on this line. In this case the form of the Benchmark-95 tasks should be further elaborated from the presentation of the results viewpoint, the contents of the tasks being exactly and strictly the same. Some optional tasks are to be integrated into the formal tasks, however. Also, the participants should deliver their results with a standard means (e.g., a 3 ½ " diskette) rather than on mixed media. In this context, the existing benchmark tasks descriptions should be revised and improved for the standardization of the results reporting for efficient processing and reproducibility. This includes both complementary information needed during the analysis of the results and due format of the reporting. The detailed design description of this benchmarking can be comprehended from the presentation of Benchmark-95 [8].

5.2. Follow-up-96: NN Training Algorithms

Although the standard back-propagation (BP) algorithm is quite popular since it is easy to understand due to its simplicity in implementation, however, there are serious limitations on the effective utilization of such algorithm (or its variants, like quick BP, batch learning and so forth), in practice. Therefore, advanced and/or new training algorithms should be used for enhanced systematic approach for network training where effective and efficient use of networks is highly required. In the present study report, the substantial improvement provided by such algorithms is clearly demonstrated with enhanced generalization and efficient i.e., fast learning capabilities. Such a benchmarking would benefit the participants to train their networks better with reference to the network limitations in practical use. In this proposal, a new benchmark data preparation is not central to the proposal as is the case in preceding proposal and Benchmark-95 data can be used for this proposal. Considering the experiences gained during the analysis of the Benchmark-95 data, such an approach is even quite appropriate. The detailed design description of this benchmarking can be comprehended from the chapter titled *Further Studies With the Benchmark Data*, above (Section 4.1) where some algorithms, suitable for the intended goals, are described.

5.3. Follow-up-96: NN Sensitivity Analysis

The sensitivity analysis is an important issue in neural network utilization since it plays important role on the performance of the network. This is because sensitivity is closely related to the gradients with respect to synaptic weights and so does the network performance so that it provides valuable information on network's status following the training. It can give information also on the pitfalls of the training process so that one might improve his/her algorithm by better understanding the functionality of the algorithm. In this context, a new condition monitoring benchmark in the form of follow-up could be designed where the gravity of the benchmark would be the sensitivity of the network in use. Such a benchmarking would benefit the participants to follow the learning process of their networks together with the limitations as well as possibilities involved. In this proposal, a new benchmark data preparation is not central to the proposal and Benchmark-95 data can be used for this proposal. With the same reasoning given in the preceding proposal, such an approach is quite appropriate. The

detailed design description of this benchmarking can be comprehended from the chapter titled *Further Analysis With the Benchmark Data*, above (Section 4.2) where some basic sensitivity studies and analyses are carried out.

5.4. Follow-up-96: NN as a Tool in Noise Analysis

In this benchmark, utilization of neural networks in noise analysis is aimed. Here, in contrast with stochastic signal processing using neural network, neural network is used to process spectral information obtained by stochastic signal processing methods. Therefore, the neural network itself is not stochastic and it is used as an information processor using the noise information as plant signatures obtained earlier. For example, the outcome of the neural network can be used for monitoring of spectral changes, or spectral interdependencies of the stochastic signals. Such applications are important in power plant operations since small spectral changes, drifts etc., can be very significant and can be the indicator of early failures. These types of failures are generally quite elusive and in the case of above example, such spectrum inspections are normally difficult to carry out on a continuous basis, in practice. The benchmark tasks in such a benchmark exercise should be seen in two steps. In the first step, noise analysis is performed to obtain noise signatures from the data appropriately. In the second step, neural network is used for knowledge acquisition using the outcomes of the preceding information and/or signal processing. Such a benchmarking is quite significant for power plant monitoring and early failure detection as a novel cooperative initiative bringing the conventional noise analysis techniques and the neural networks together.

5.5. Follow-up-96: NN for Reactor Noise Analysis with Stochastic Signals

Stochastic neural networks are especially important for processing stochastic signals and they are of main interest for parametric and non-parametric reactor noise analysis studies with neural networks. Additionally, they are important especially for stochastic system dynamics modeling by neural networks. The neural network in such applications can be used for both system and signal modeling as a novel stochastic information-processing tool. For basic understanding and utilization neural networks with stochastic signals by the help of basic feed-forward type structures a pioneer benchmarking would be a very important initiative in this direction. Presumably, such a benchmark would be most appreciable for the reactor noise community among the other proposals. Stochastic neural networks generally require stochastic training algorithms because of required fast convergence during adaptation with stochastic signals. Kalman filtering training algorithm is the most effective training algorithm in this respect. An example neural network approach for processing stochastic signals using *wavelets* is described before [9]. The detailed design description of this benchmarking can easily be comprehended concerning stochastic parameter estimation, signal prediction, and spectral estimation problems in the field of noise analysis. Some software support can be provided in this benchmark due to need for advanced training algorithms relative to back-propagation.

6. CONCLUSIONS

As a new field, the scope of neural network research is wide and subject to exploration. Even with the limited scope of Benchmark-95, the present analysis results & follow-up'95 report at hand, in addition to its mission, presents important research results on neural networks due to the nature of its mission. This is provided by both participants through their contributions and studies at hand here performed for thorough evaluation of these contributions. With the capacity of having the present results of the benchmark analysis, it is to conclude that, the Benchmark-95 has received a satisfactory attention and feedback from the participants. It provided the neural network users in the nuclear industry with a valuable information source to test their neural-network-based methods for their diverse interests spanning nuclear industry and Nuclear Science & Technology teaching. In addition to rather common nature of operational plant data, in particular, the two peculiar operations in the data, i.e., *rinsing and stretch-out* provide one with important test facilities to verify and/or validate the performance of a neural network. In the benchmark design, these are especially included and some benchmark tasks are intentionally so designed that each involves either of these operational cases for different learning and/or recall phases in order to be able evaluate the benchmark tasks execution performances from different viewpoints. The benchmark data expectedly will preserve its value and will continue to be subject to further analyses beyond the benchmark analyses for experimental research on neural networks in nuclear industry.

The benchmark undoubtedly helped to the participants to gain insight into the properties of neural networks concerning their functionality, usefulness and utilization, the latter especially being concerned with nuclear industry. With reference to the industrial neural network implementations, the neural network technology closely associates with the nuclear technology as this can easily be noted from the literature in the year 90s. However, for the recognition of this technology for critical applications, like nuclear power plant fault detection, diagnosis, monitoring for instance, apparently requires further maturity of this technology. In this respect, the present benchmark also gave some key indications of the potential role of neural networks in nuclear industry. These indications revealed the importance of the establishment of some general guidelines for enhanced neural network training and effective utilization. Referring to this, a follow-up benchmark will play an important role further. The benchmark has also provided the opportunity of exchanging the neural network experiences among the participants and others. It highlighted the potentialities of neural network technology in nuclear technology during SMORN-VII symposium and conveyed a message to the symposium participants. In the light of these considerations and conclusions, a bottom line would be the strong recommendation of a follow-up of Neural Network Benchmark-95 (SMORN-VII). This would be very beneficial for better comprehension and effective utilization of neural networks in nuclear industry eventually leading to benefit nuclear technology, from the potentialities of neural networks and the related technology.

REFERENCES

1. E. Türkcan et al., *Information processing system and neural network utilization for accident management support*, Proc. FISA-95 Symposium, EU Research on Severe Accidents, Luxemboutg, 20-22 November 95. Also appeared in Report, Netherland Energy Research Foundation, ECN-RX--94-031, May 1994, The Netherlands

2. E.Türkcan, Ö. Ciftcioglu and K. Nabeshima, *Neural networks for real-time NPP monitoring*, Nuclear Europe Worldscan 11-12 November/December 1993

3. Ö. Ciftcioglu and E. Türkcan, *On-line plant-wide monitoring using neural networks*, 8th Power Plant Dynamics, Control and Testing Symposium, May 27-29, 1992, Knoxville, Tennessee, (USA)

4. Ö. Ciftcioglu and E. Türkcan, *Neural network training by parameter optimization approach*, Proc. International Conference on Artificial Intelligence ICANN'93, Amsterdam, September 13-16,1993, (The Netherlands)

5. Ö. Ciftcioglu and E. Türkcan, *Optimal training for neural networks applied to nuclear technology*, Neural Networks: Artificial Intelligence and Industrial Applications, Proc. 3rd Annual SNN Symposium of Neural Networks, 14-15 September,Nijmegen, (The Netherlands), Eds. Bert Kappen and Stan Gielen, Springer Verlag ISBN 3-540-19992-6

6. Ö. Ciftcioglu and E. Türkcan, *Adaptive training of feed-forward neural networks by Kalman filtering*, Report, Netherland Energy Research Foundation, ECN-R--95-001, February 1995, (The Netherlands)

7. OECD-NEA/NSC/DOC(96)17, May 1996, *OVERVIEW*, 7th Symposium on Reactor Surveillance and Diagnostics (SMORN -VII), 19th to 23rd June 1995, Avignon (France)

8. E. Türkcan and Ö. Ciftcioglu, *Neural Network Benchmark for SMORN-VII*, (Extended Summary Report), Proc. 7th Symposium on Reactor Surveillance and Diagnostics (SMORN-VII), 19-24 June, 1995, Avignon (France)

9. Ö. Ciftcioglu and E. Türkcan, *Wavelet Understands Neural Networks*, Proc. IEEE 2nd Signal Processing Symposium, 8-9 April 1994, Gokova, Turkey. Also appeared in Report, Netherlands Energy Research Foundation, ECN-RX--94-031, May 1994, (The Netherlands)

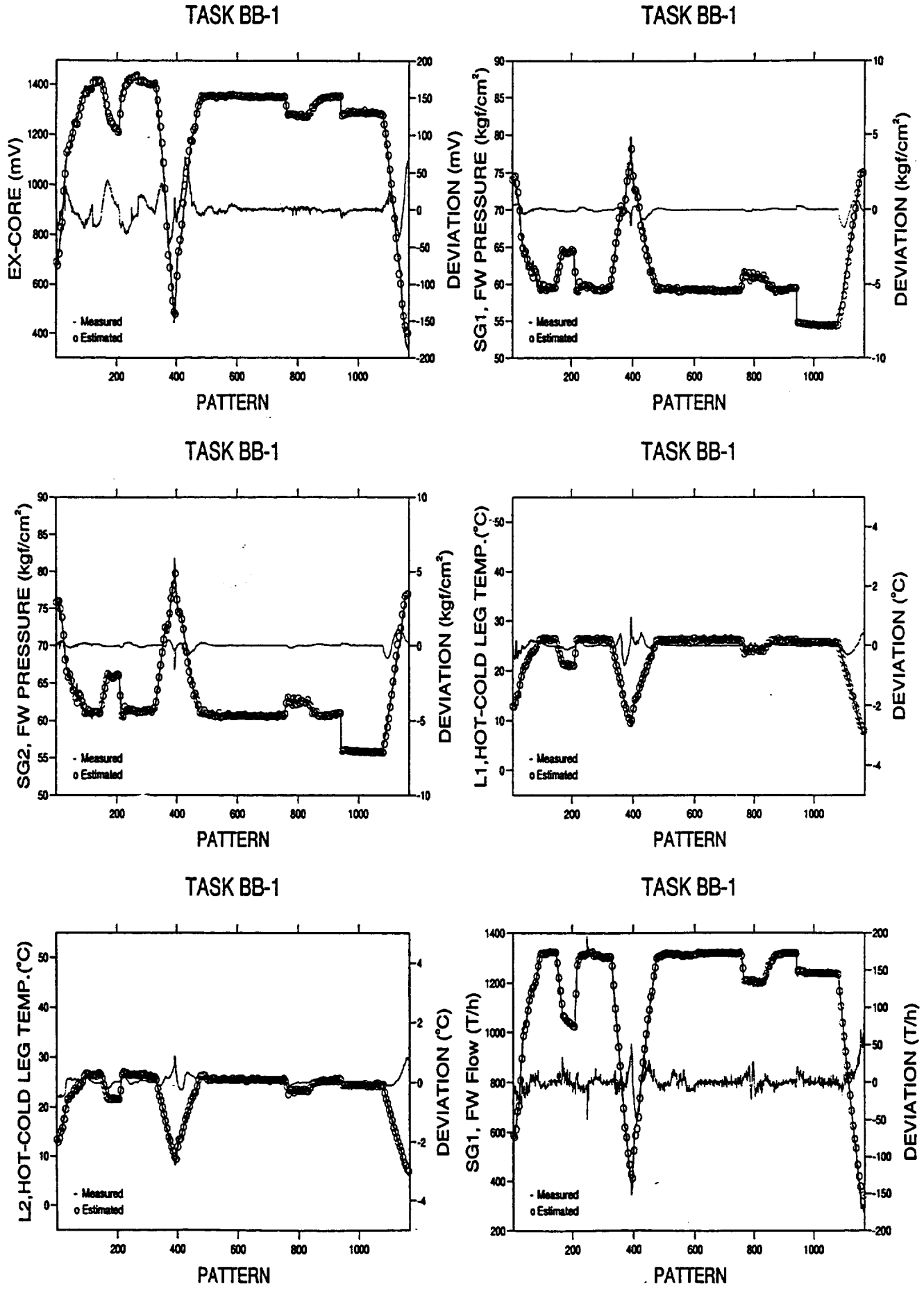

Fig.6: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is standard backpropagation. For training 1000 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.

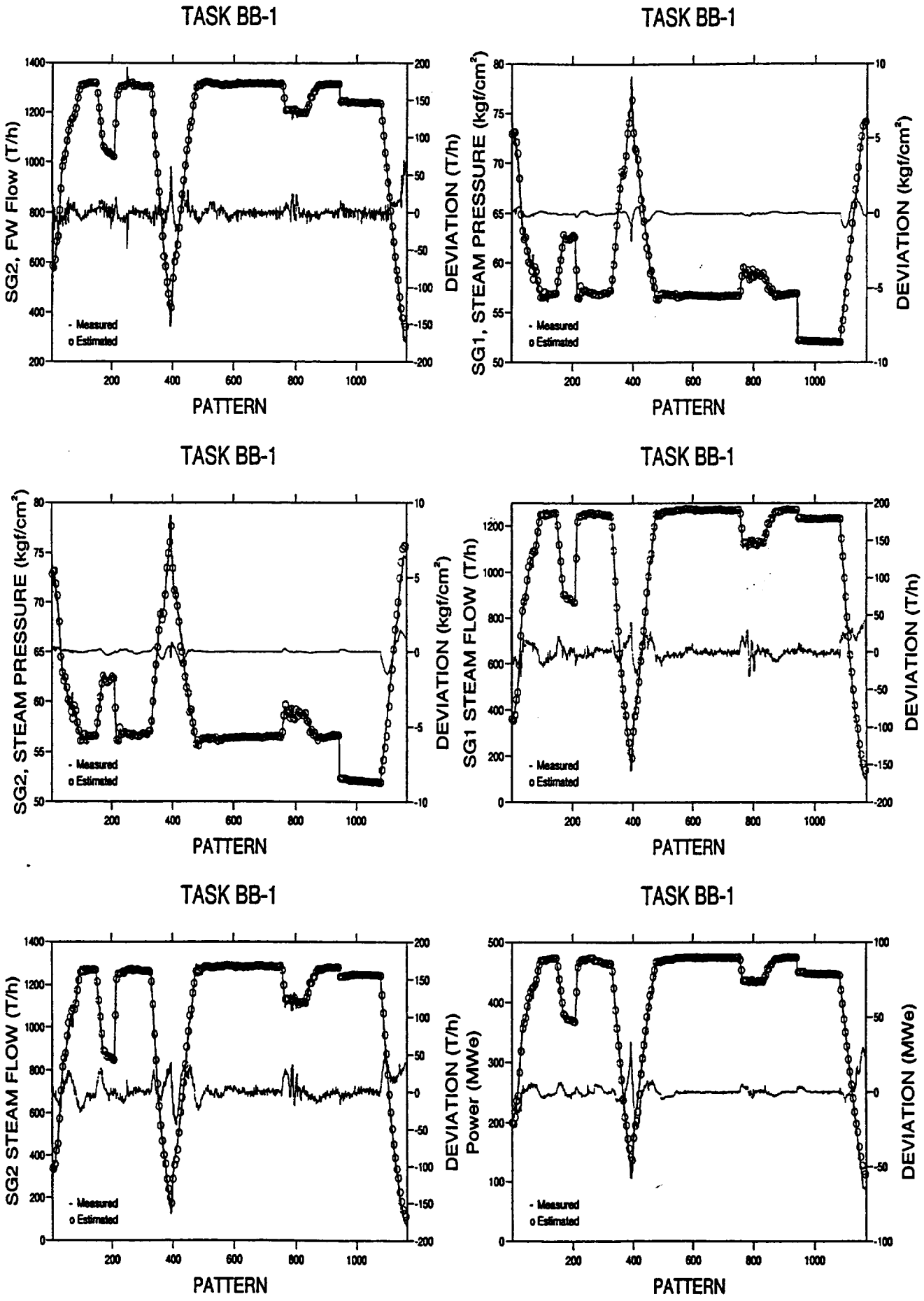

Fig.6 (continued)

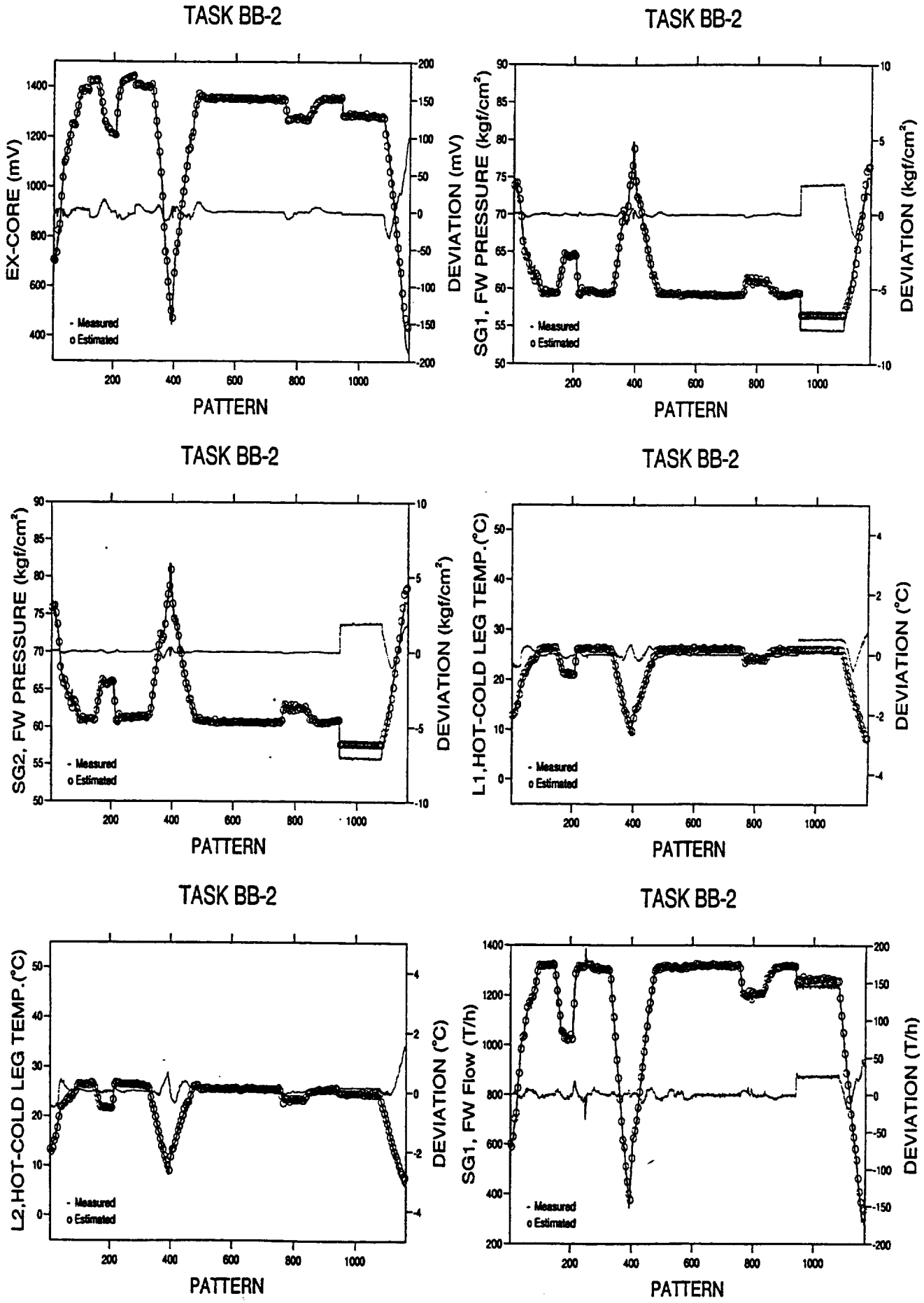

Fig.7: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is standard backpropagation. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.

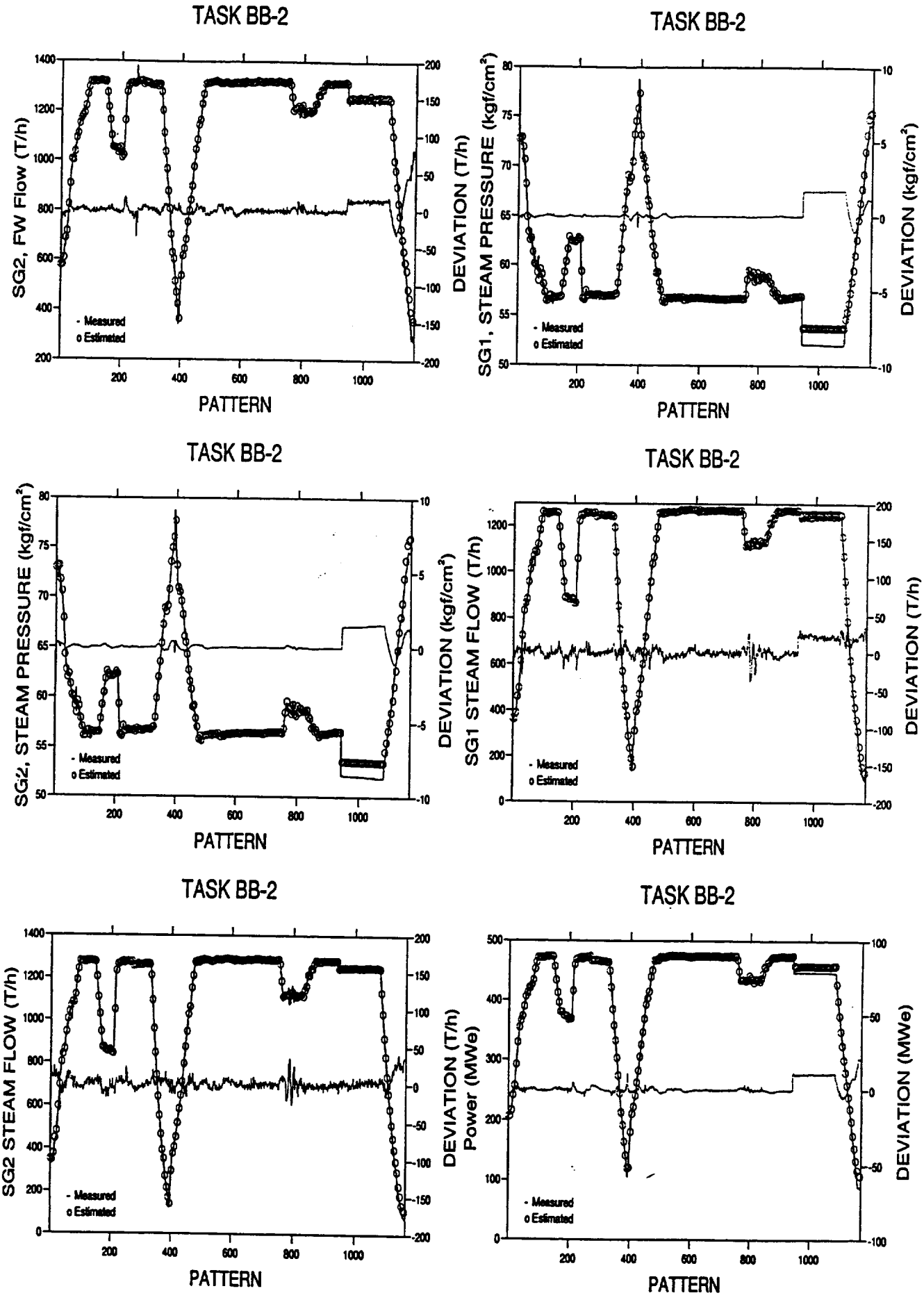

Fig.7 (continued)

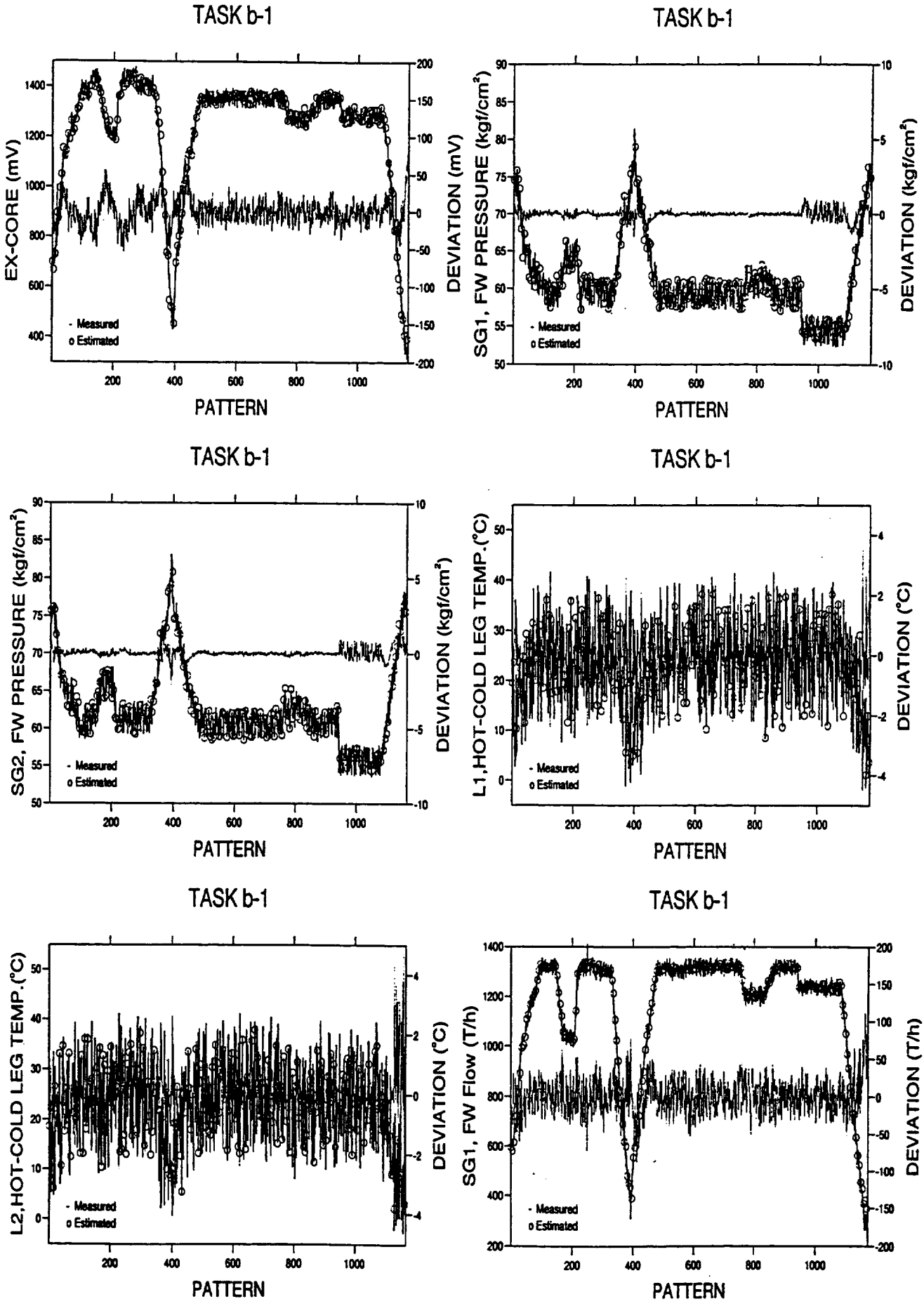

Fig.8: The benchmark data with additive noise. The standard deviation of the noise is 5% of the maximum value of the signal value among the total patterns. The standard deviation is kept constant for the respective signal throughout the patterns. The error level is irrelevant (s. the text).

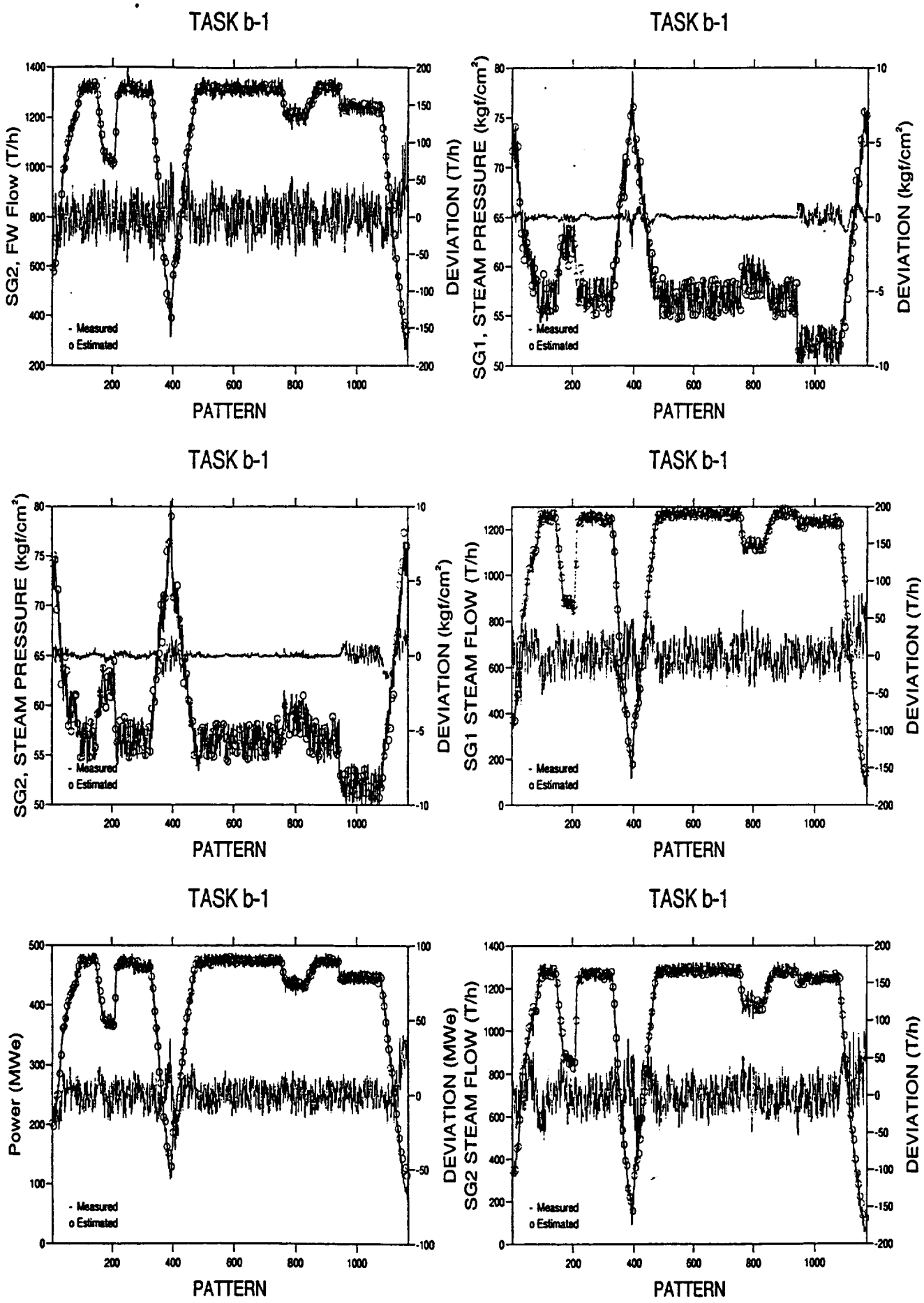

Fig.8 (continued)

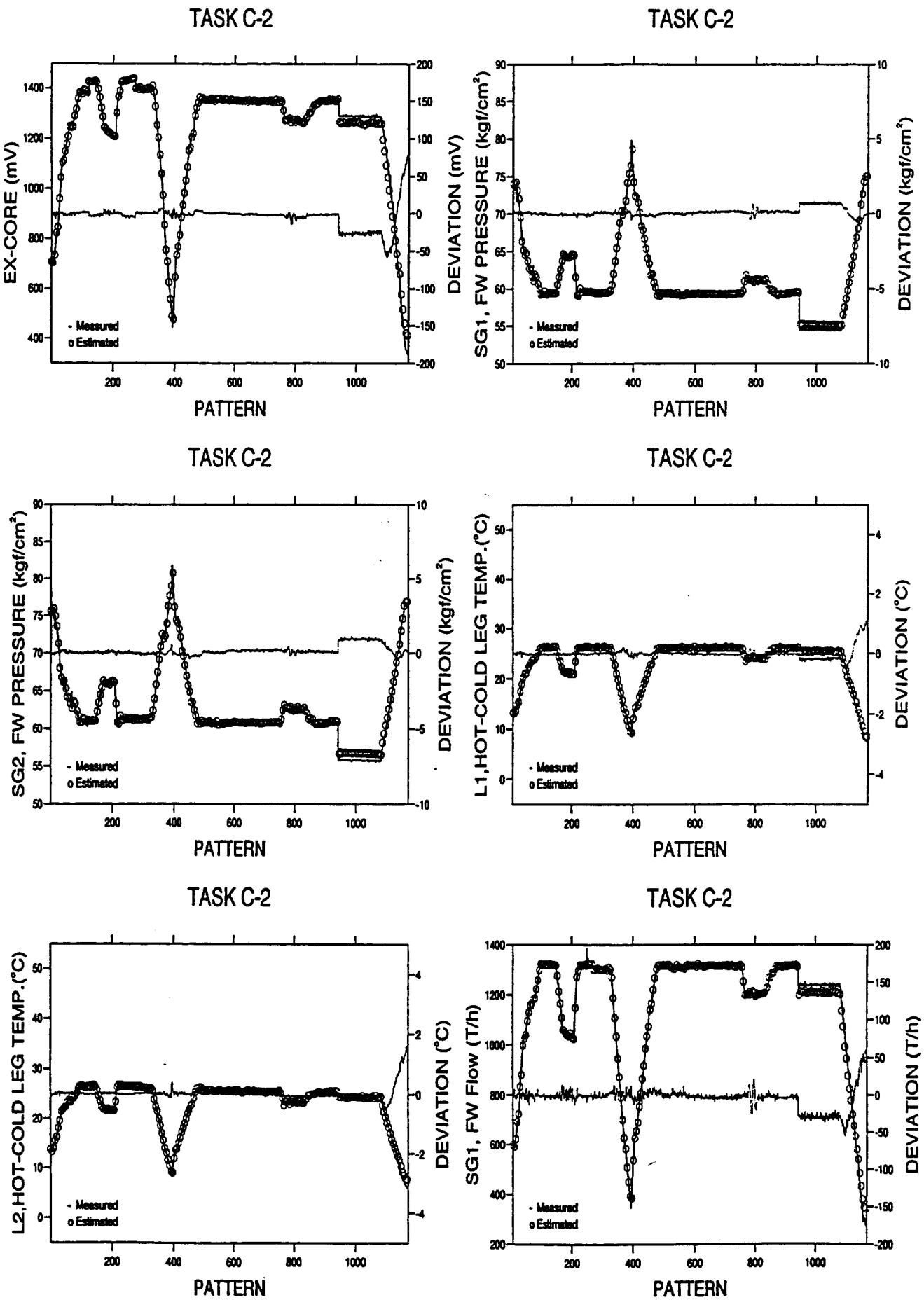

Fig.9: Processing the benchmark data . The network structure is autoassociative and training method is variable-metric optimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.

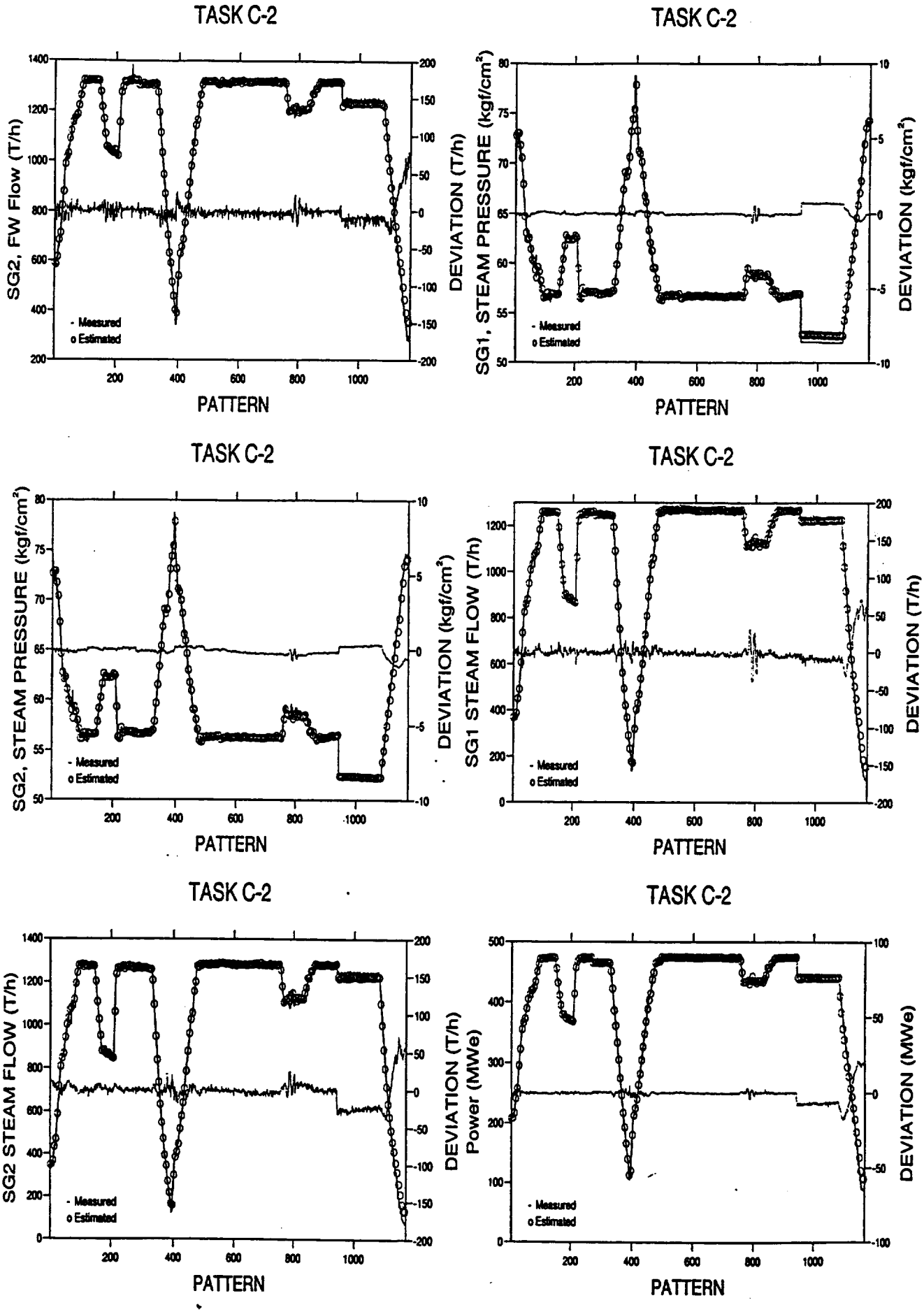

Fig.9 (continued)

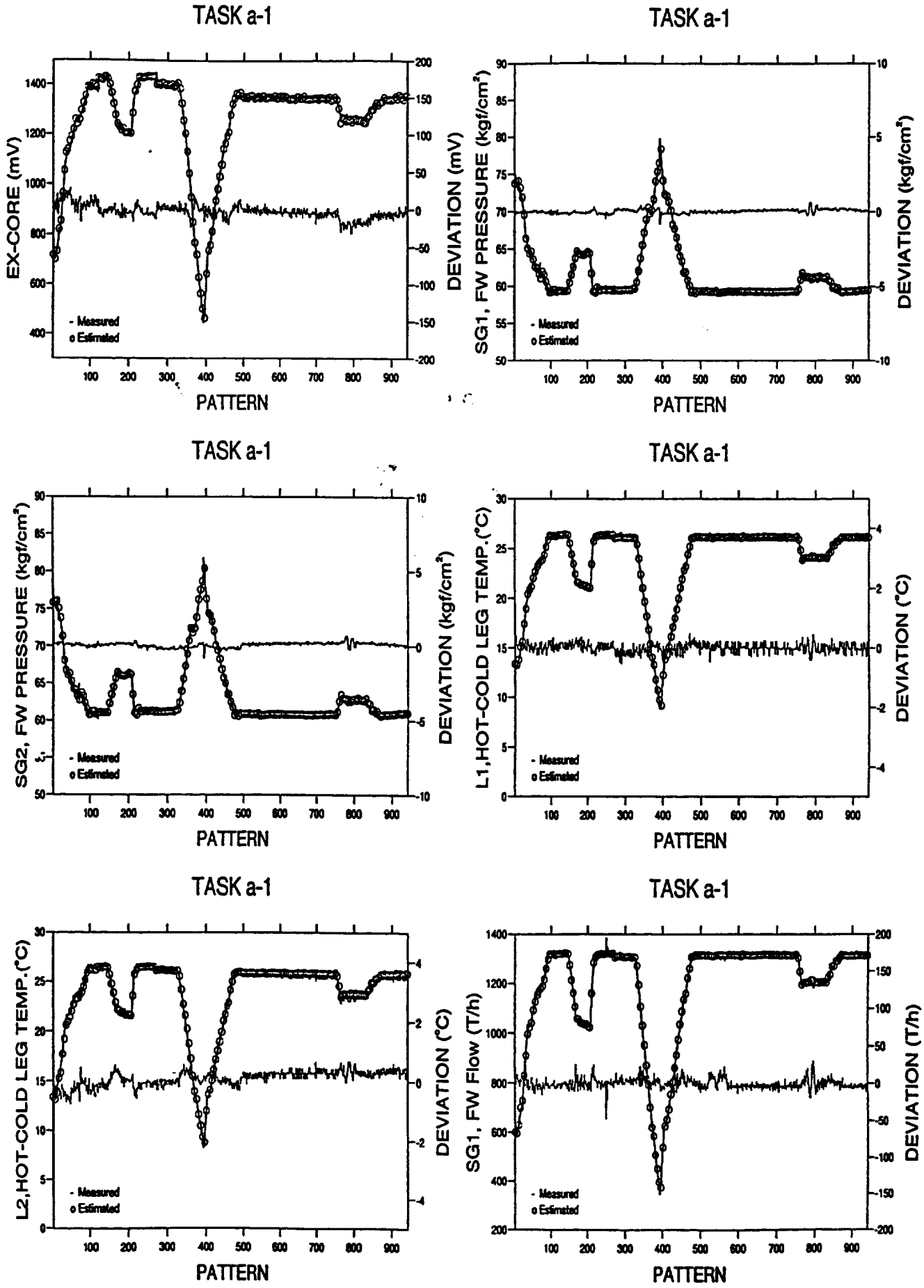

Fig.10: Processing the benchmark data The network structure is autoassociative and the training method is standard backpropagation. For training 600 patterns from the beginning is used. The recall is performed over 942 patterns from the beginning [7].

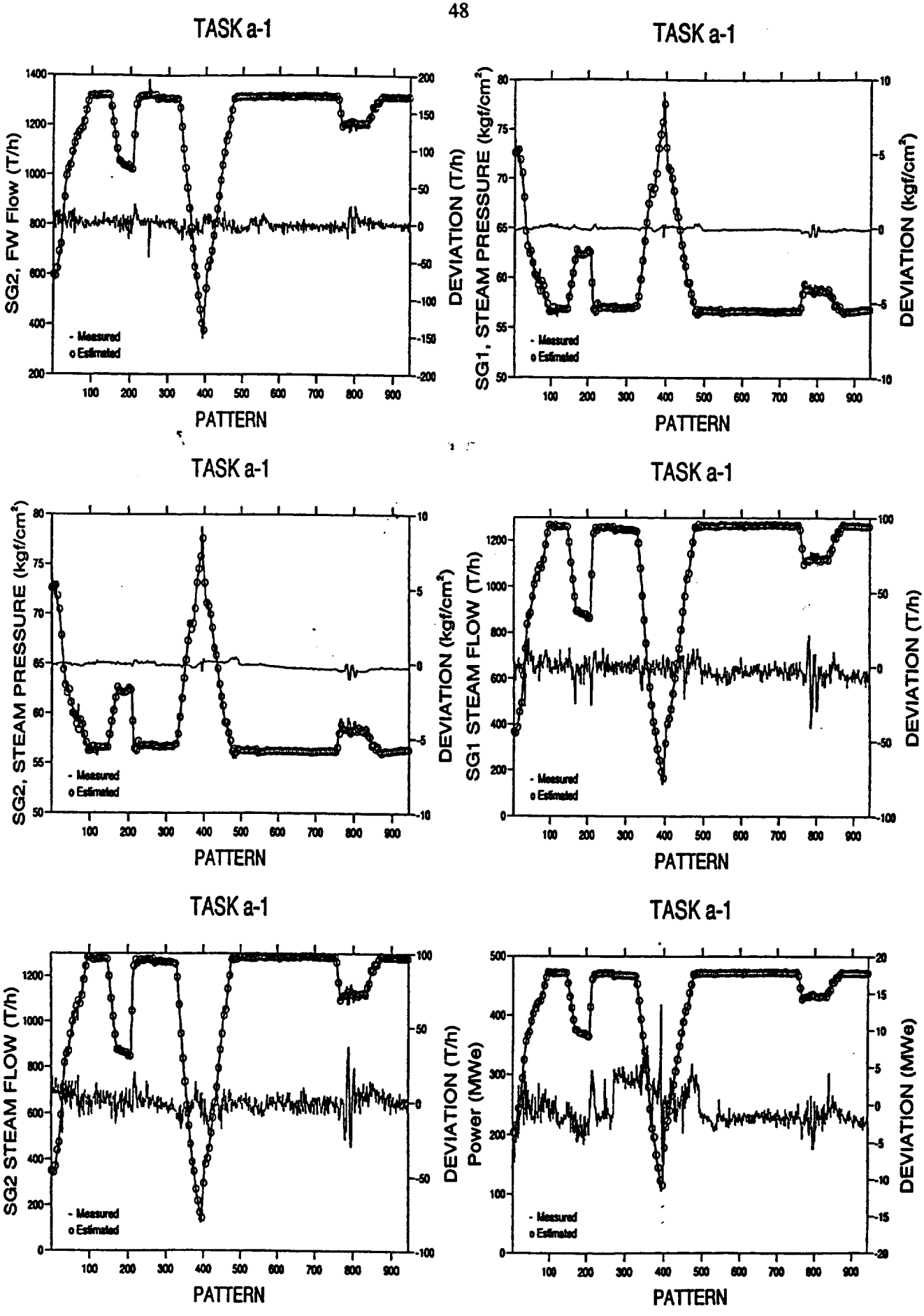

Fig.10 (continued)

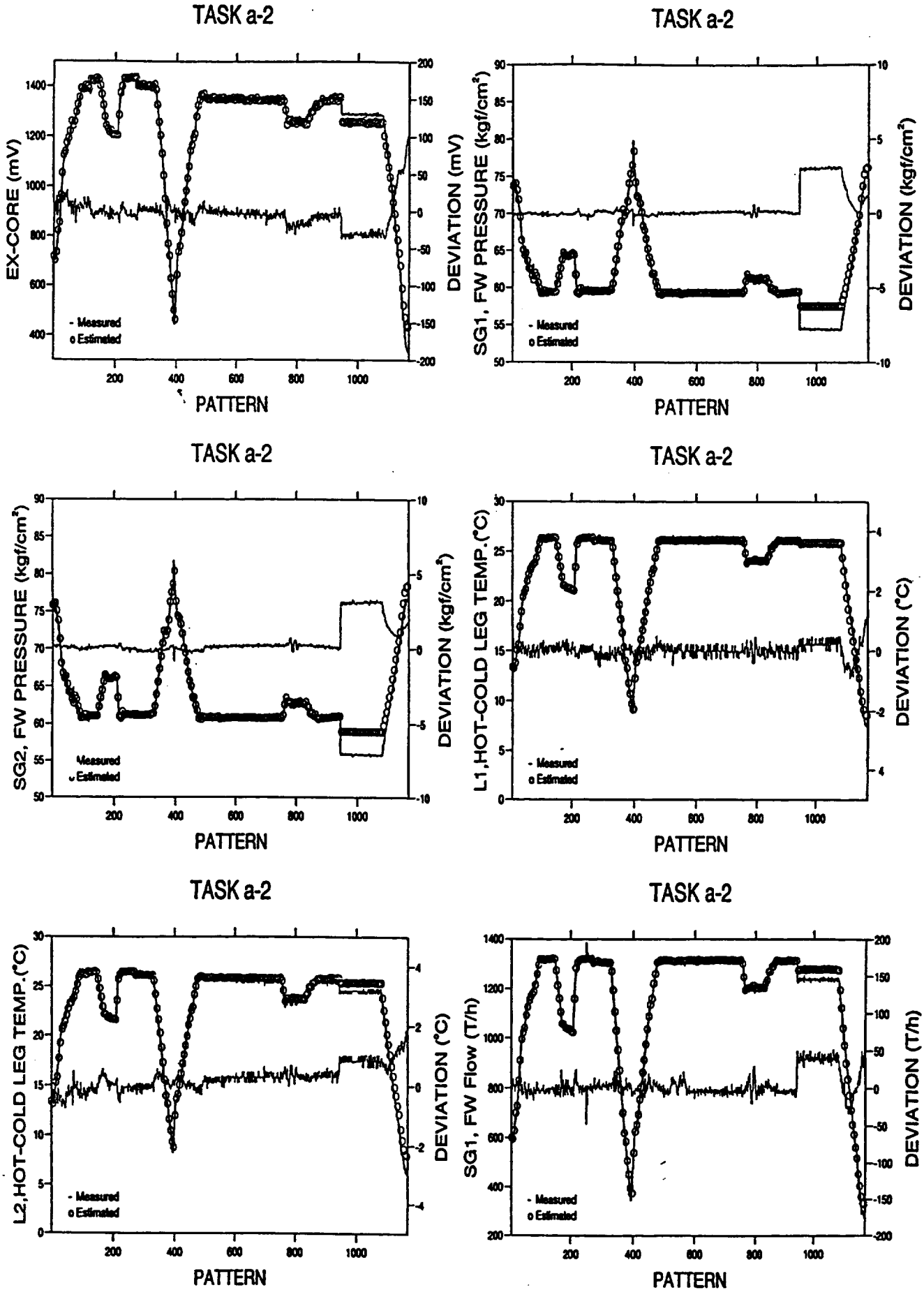

Fig.11: Processing the benchmark data The network structure is autoassociative and the training method is standard backpropagation. For training 942 patterns from the beginning are used. The recall is performed over 1169 patterns from the beginning [7].

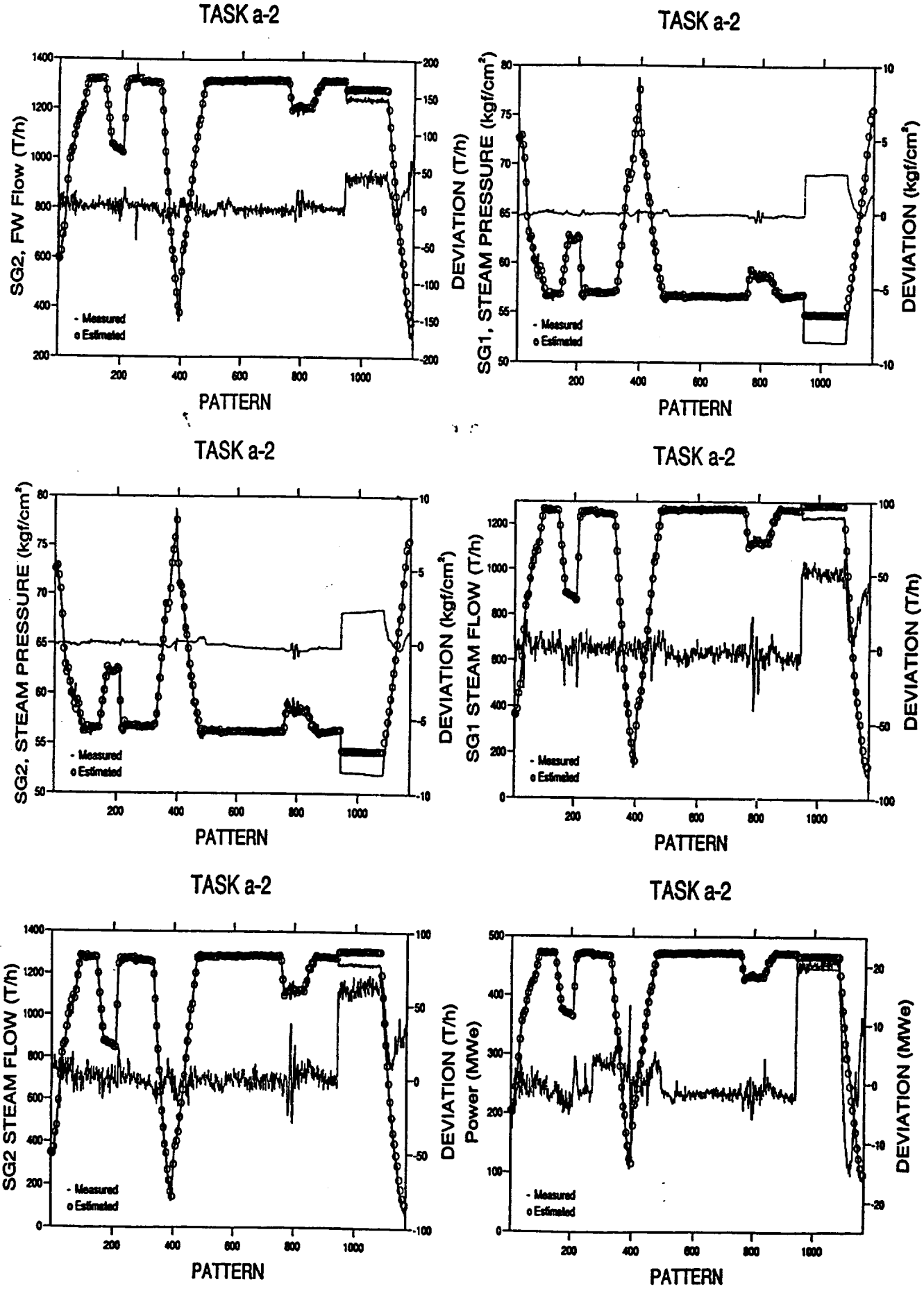

Fig.11 (continued)

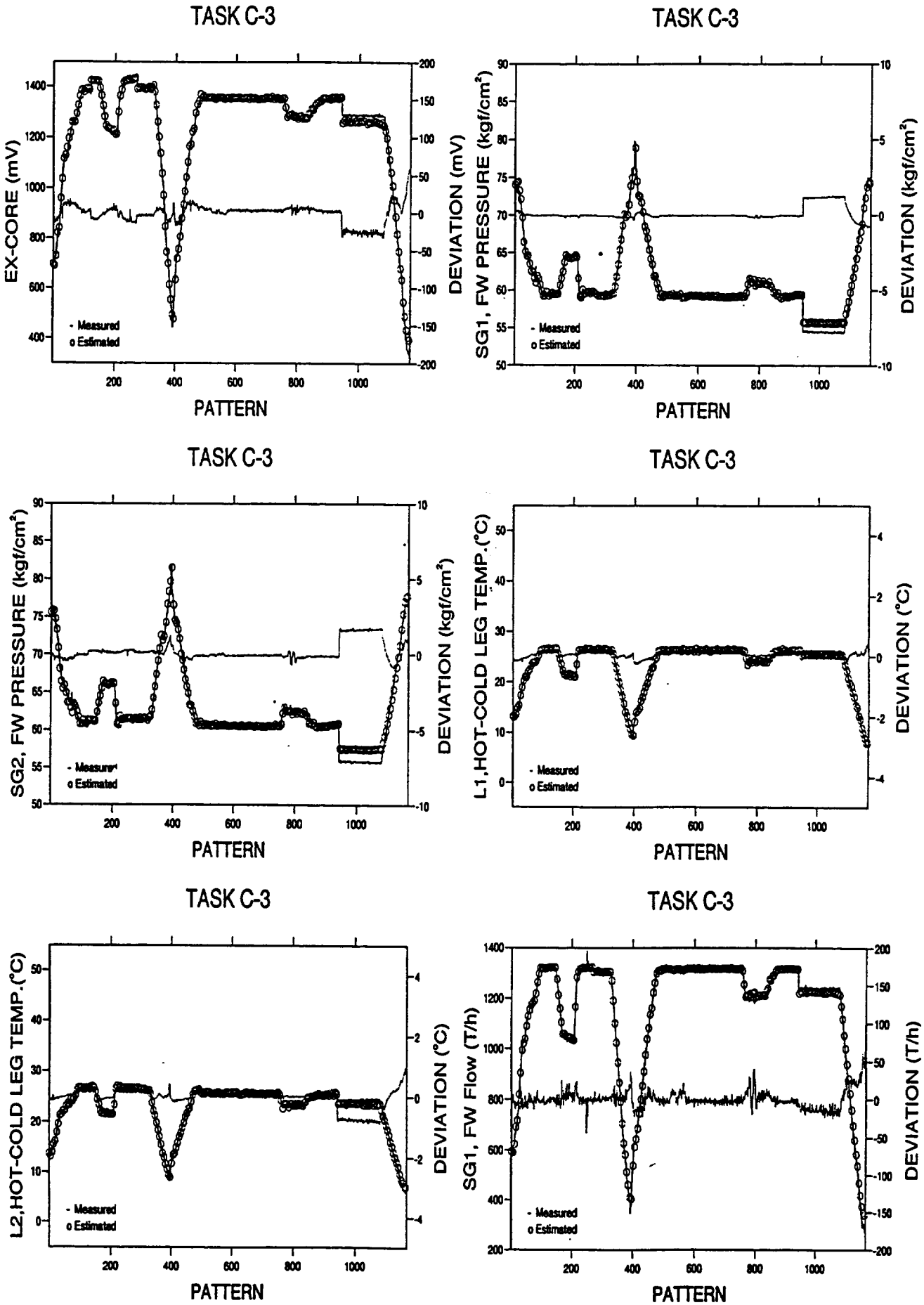

Fig.12:Processing the benchmark data with additive noise.The network structure is autoassociative and training method is variable-metric optimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is1169.

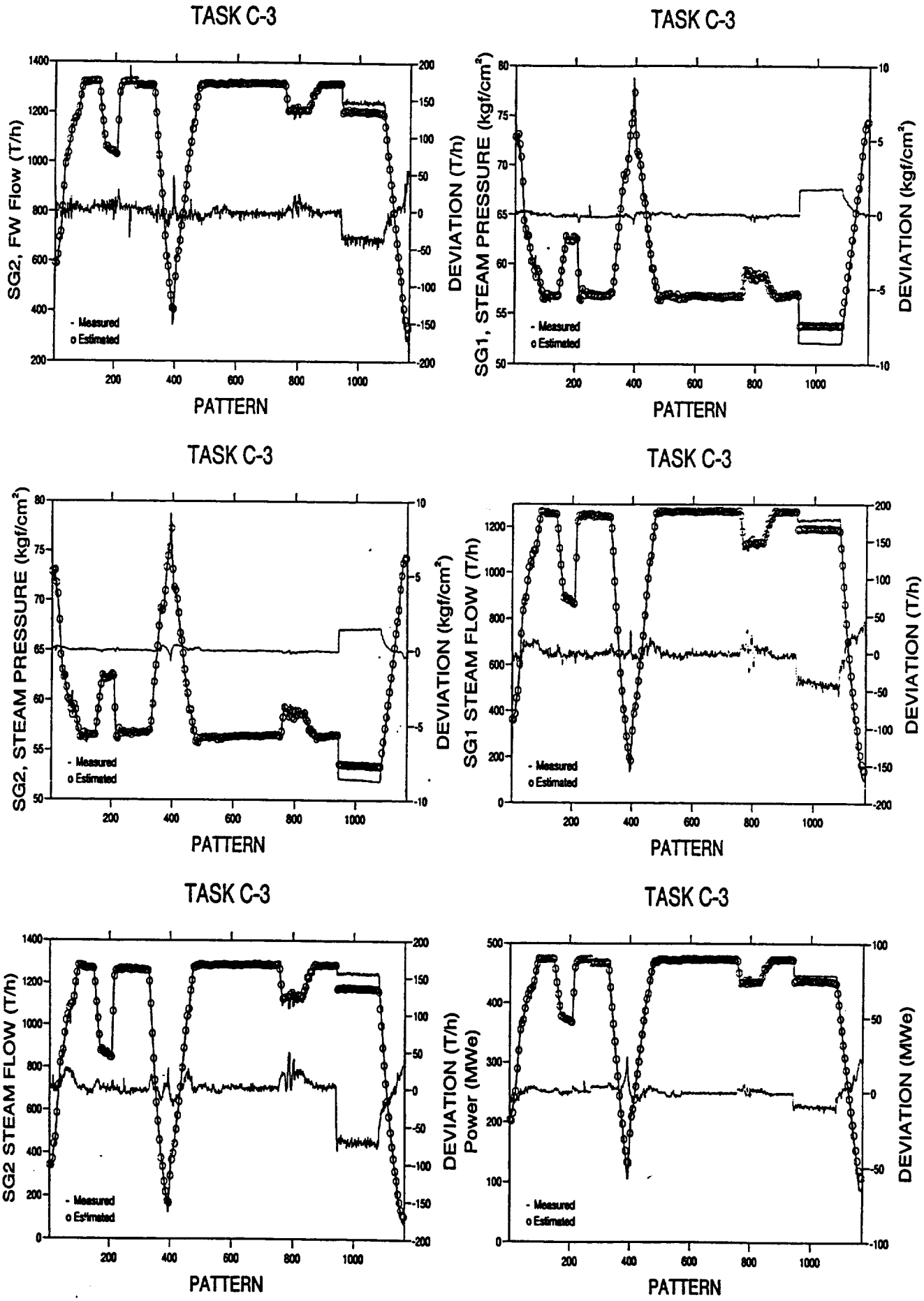

Fig12 (continued)

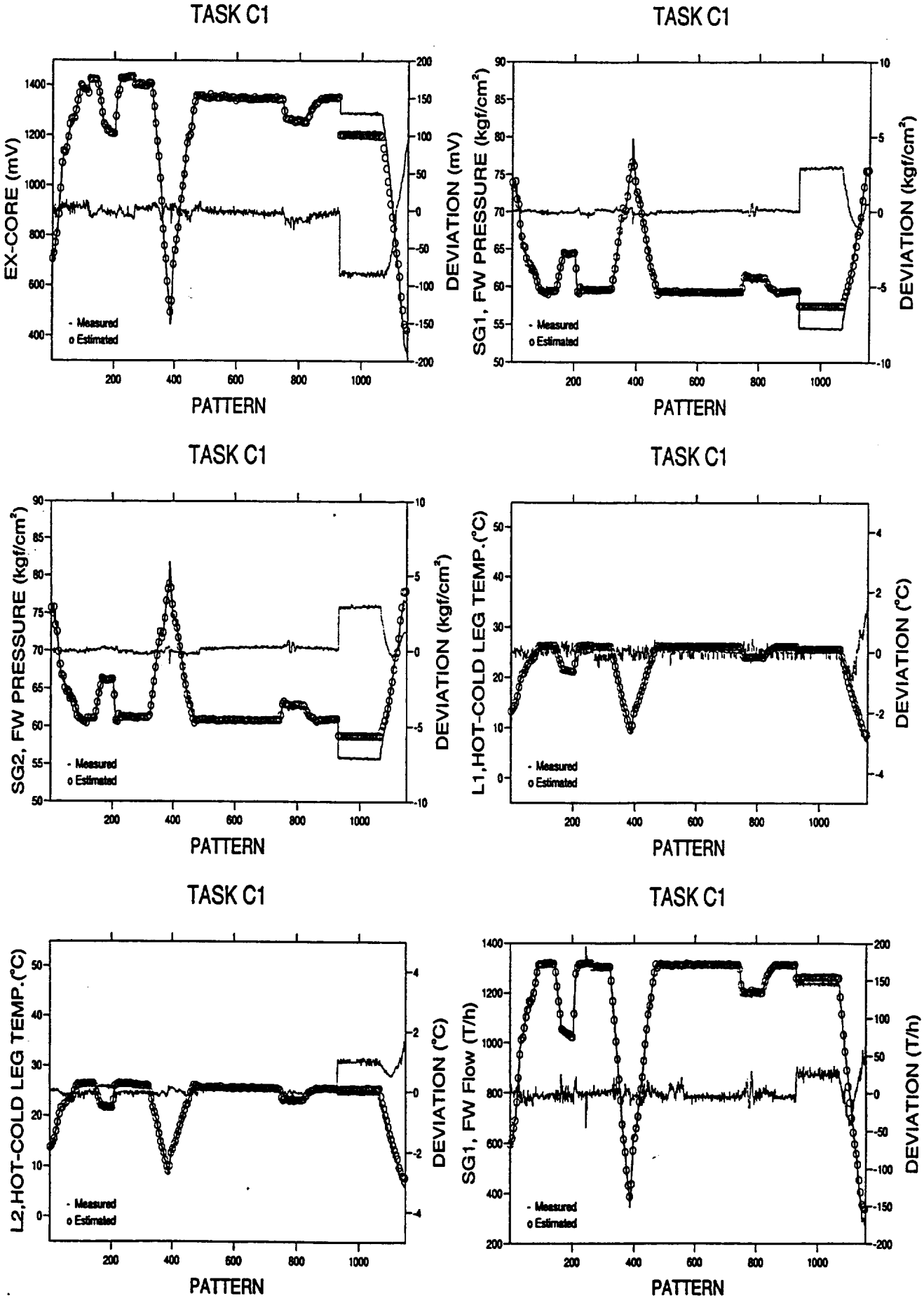

Fig.13:Processing the benchmark data with additive noise.The network structure is autoassociative and training method is standard backpropagation. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is1169.

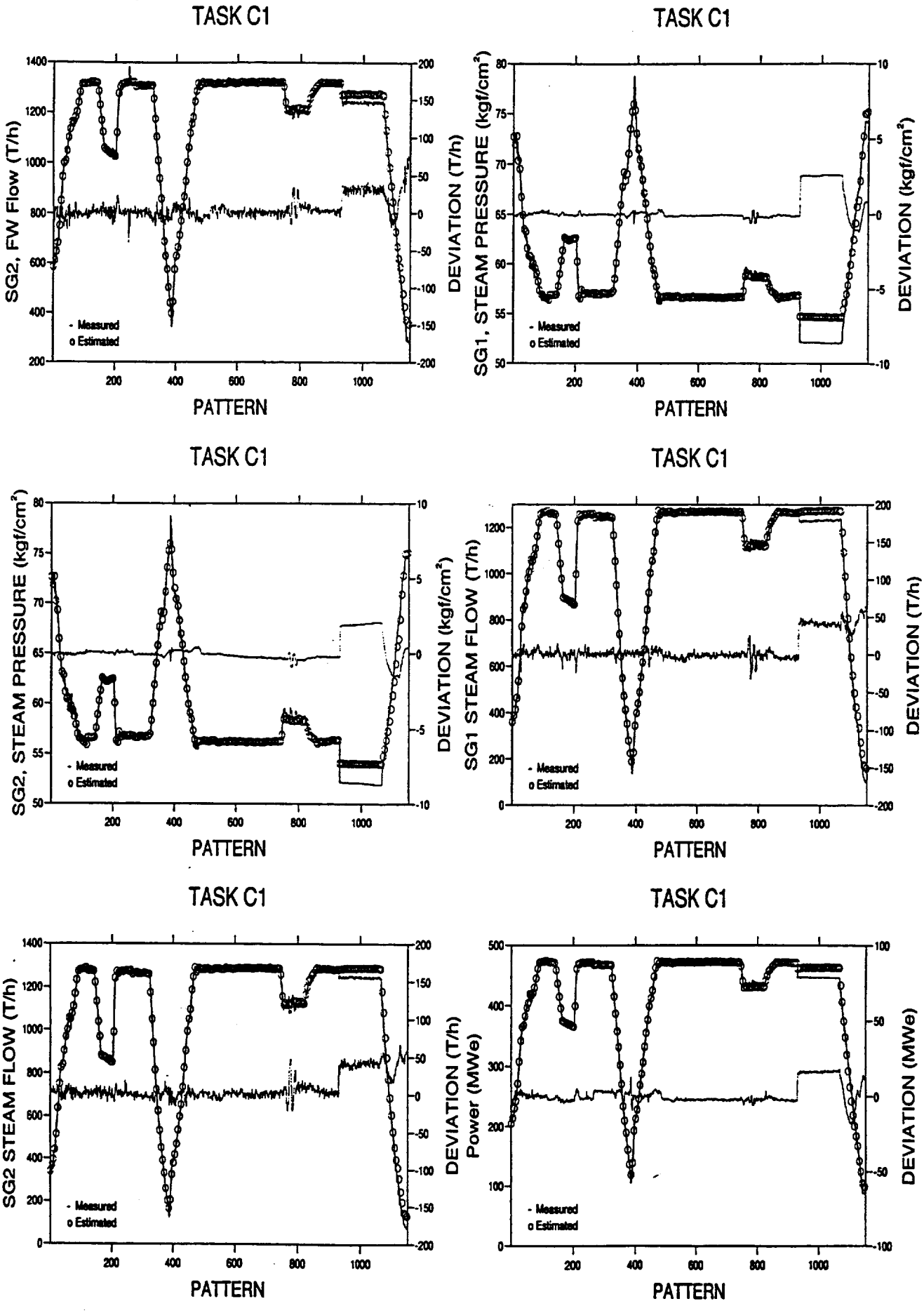

Fig.13 (continued)

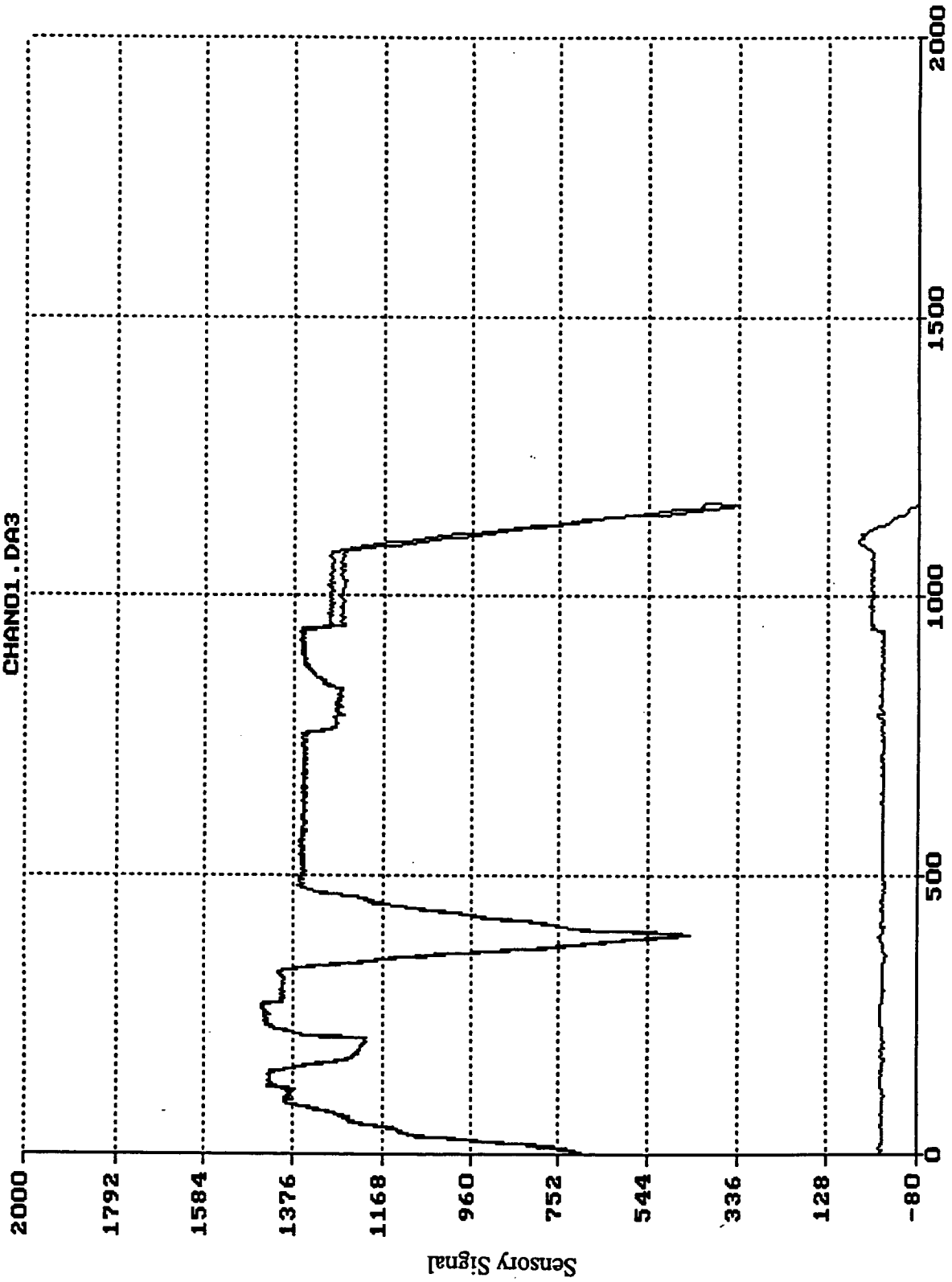

Fig.14: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of pátterns which is 1169. The lowermost graph indicates the error involved. The analysis is carried out for signal #1.

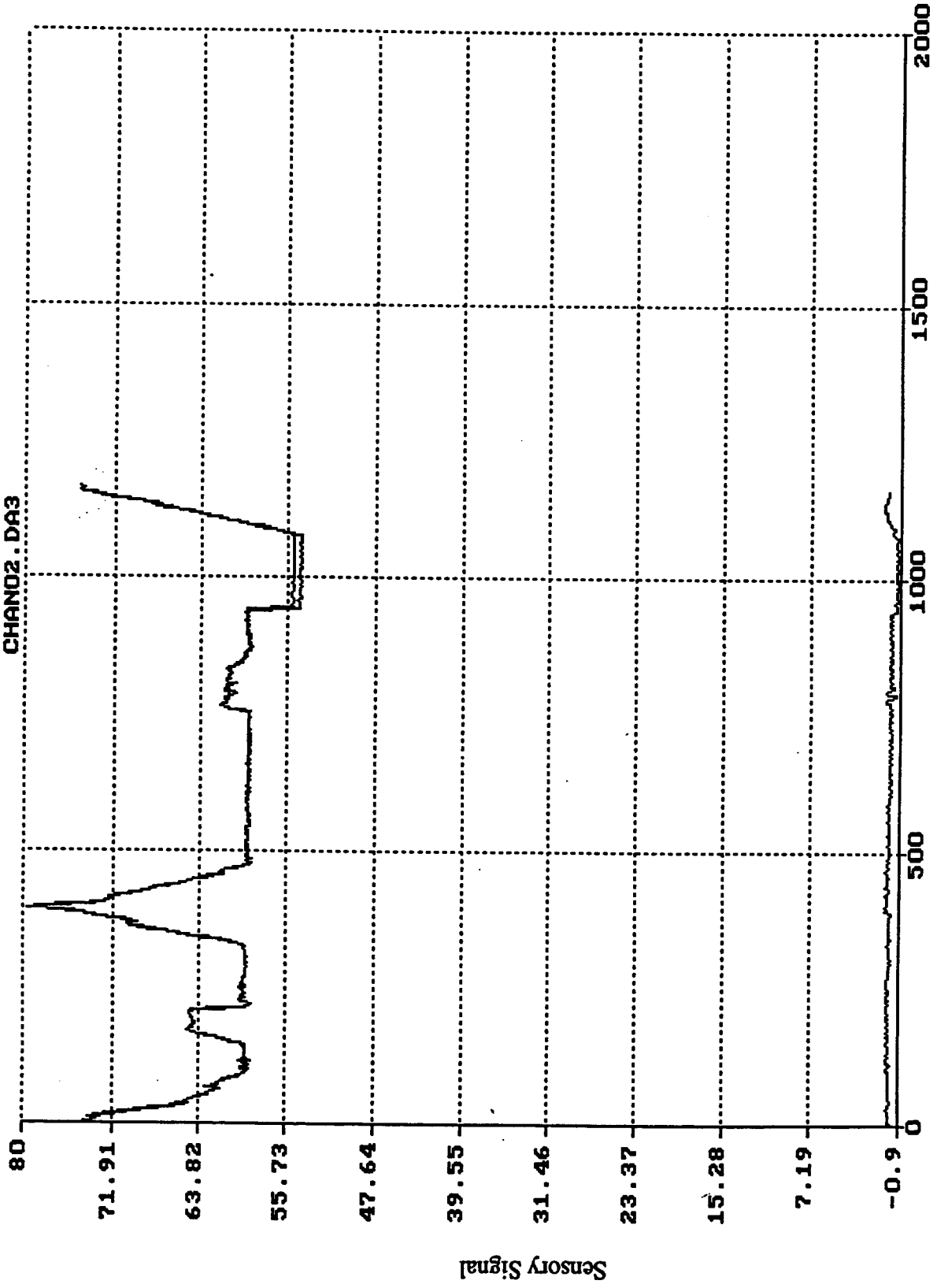

Fig.15: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.
The lowermost graph indicates the error involved. The analysis is carried out for signal #2.

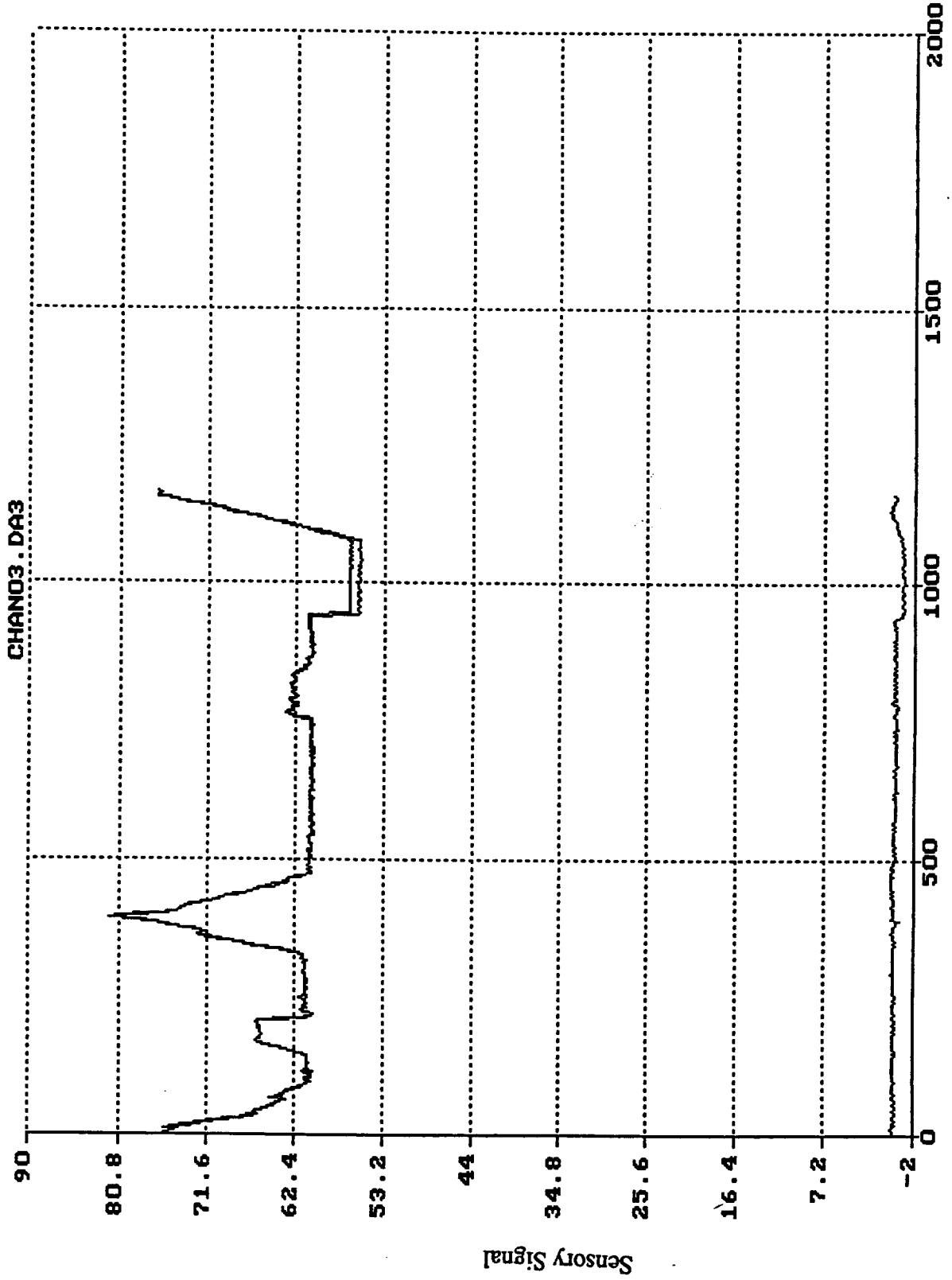

Fig.16: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.
The lowermost graph indicates the error involved. The analysis is carried out for signal #3.

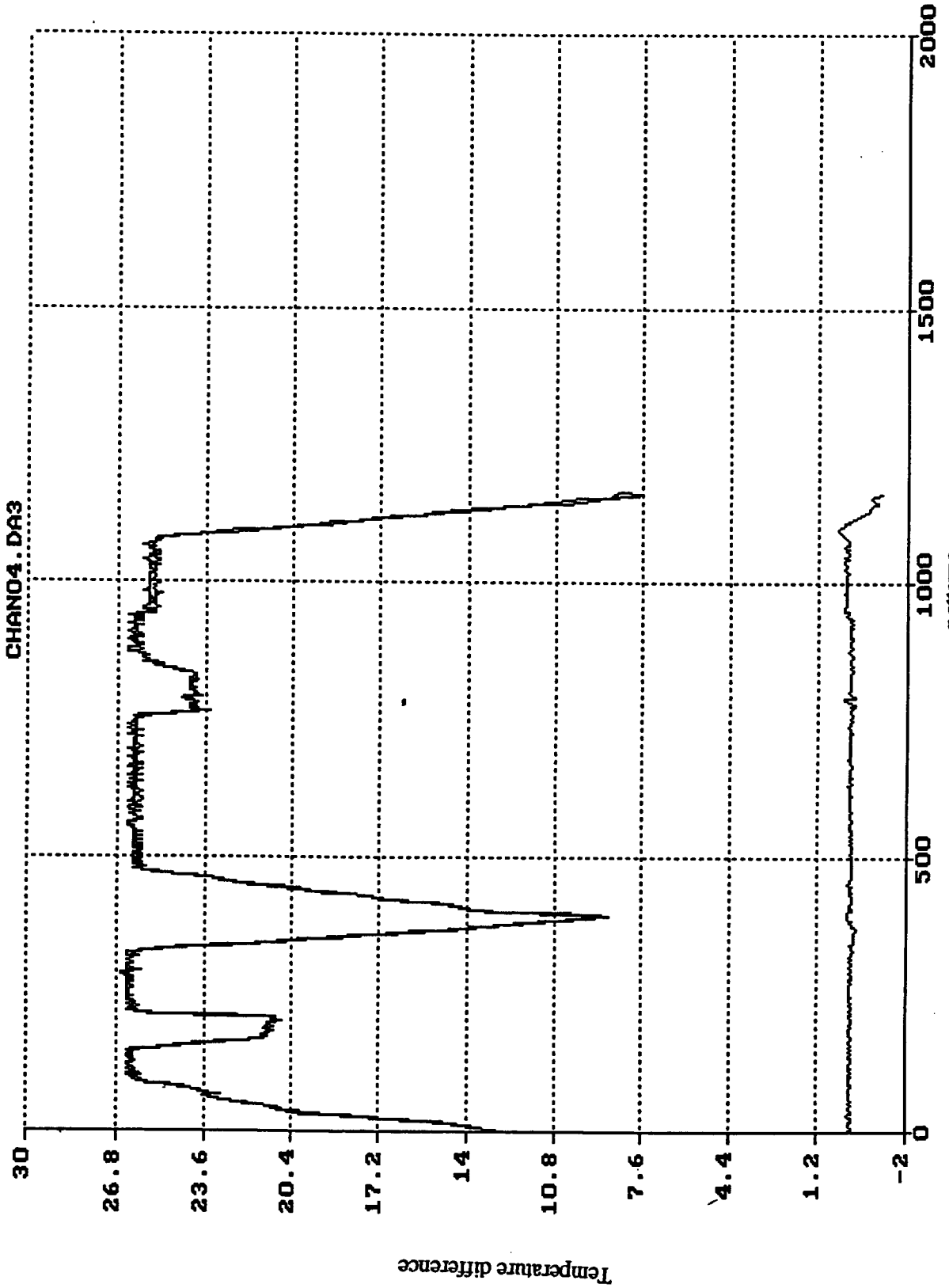

Fig.17: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.
The lowermost graph indicates the error involved. The analysis is carried out for signal #4.

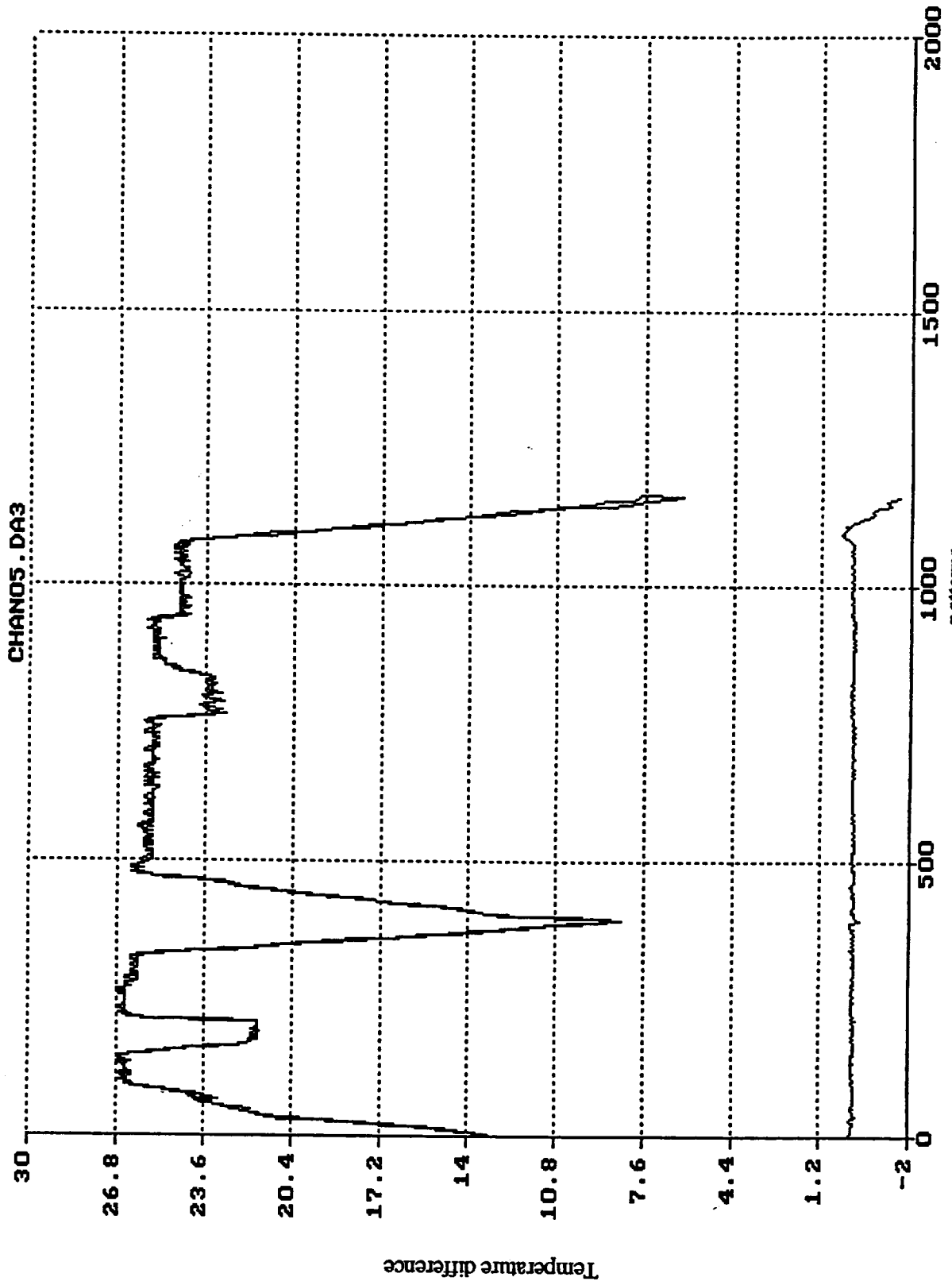

Fig.18: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.
The lowermost graph indicates the error involved. The analysis is carried out for signal #5.

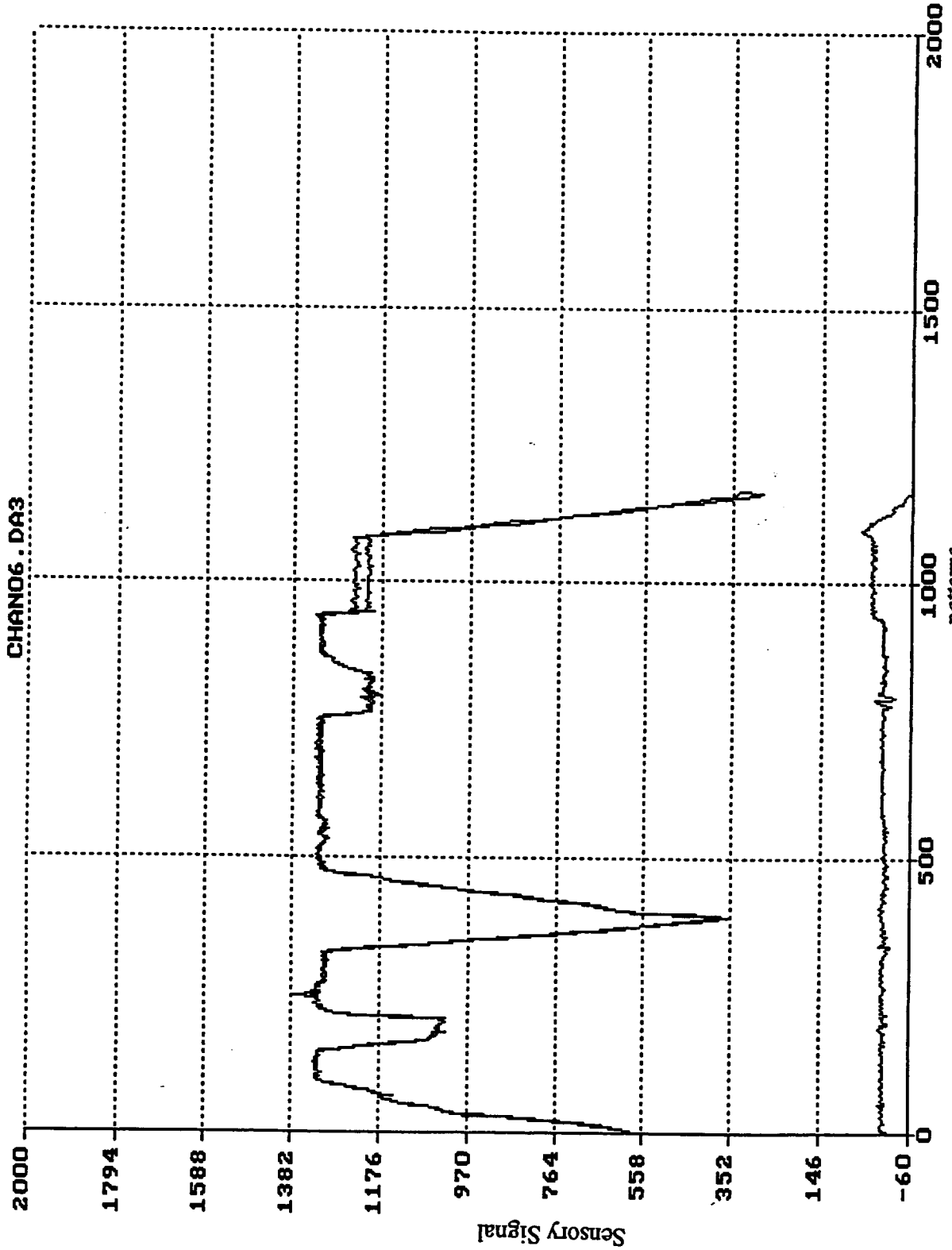

Fig.19: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169. The lowermost graph indicates the error involved. The analysis is carried out for signal #6.

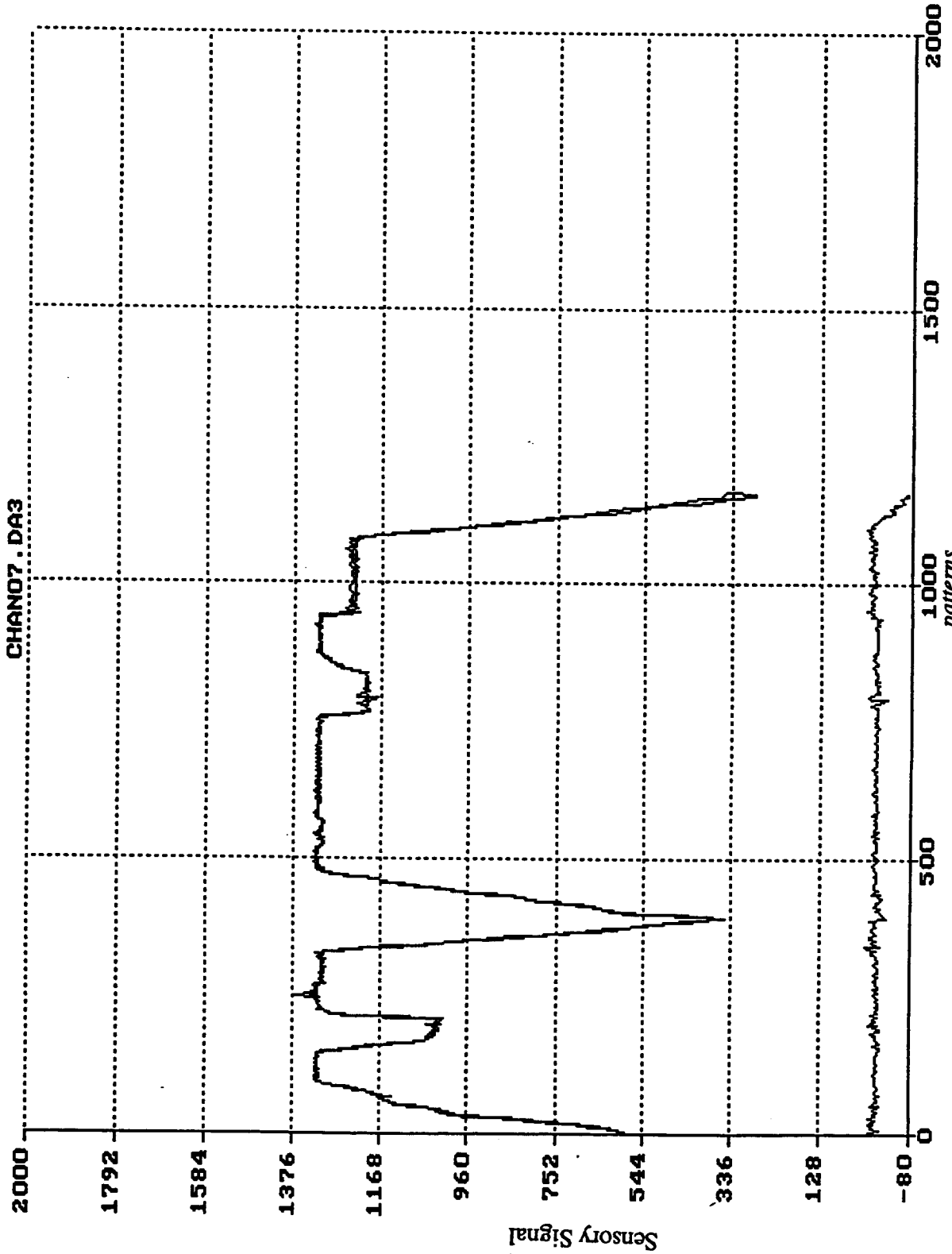

Fig.20: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.
The lowermost graph indicates the error involved. The analysis is carried out for signal #7.

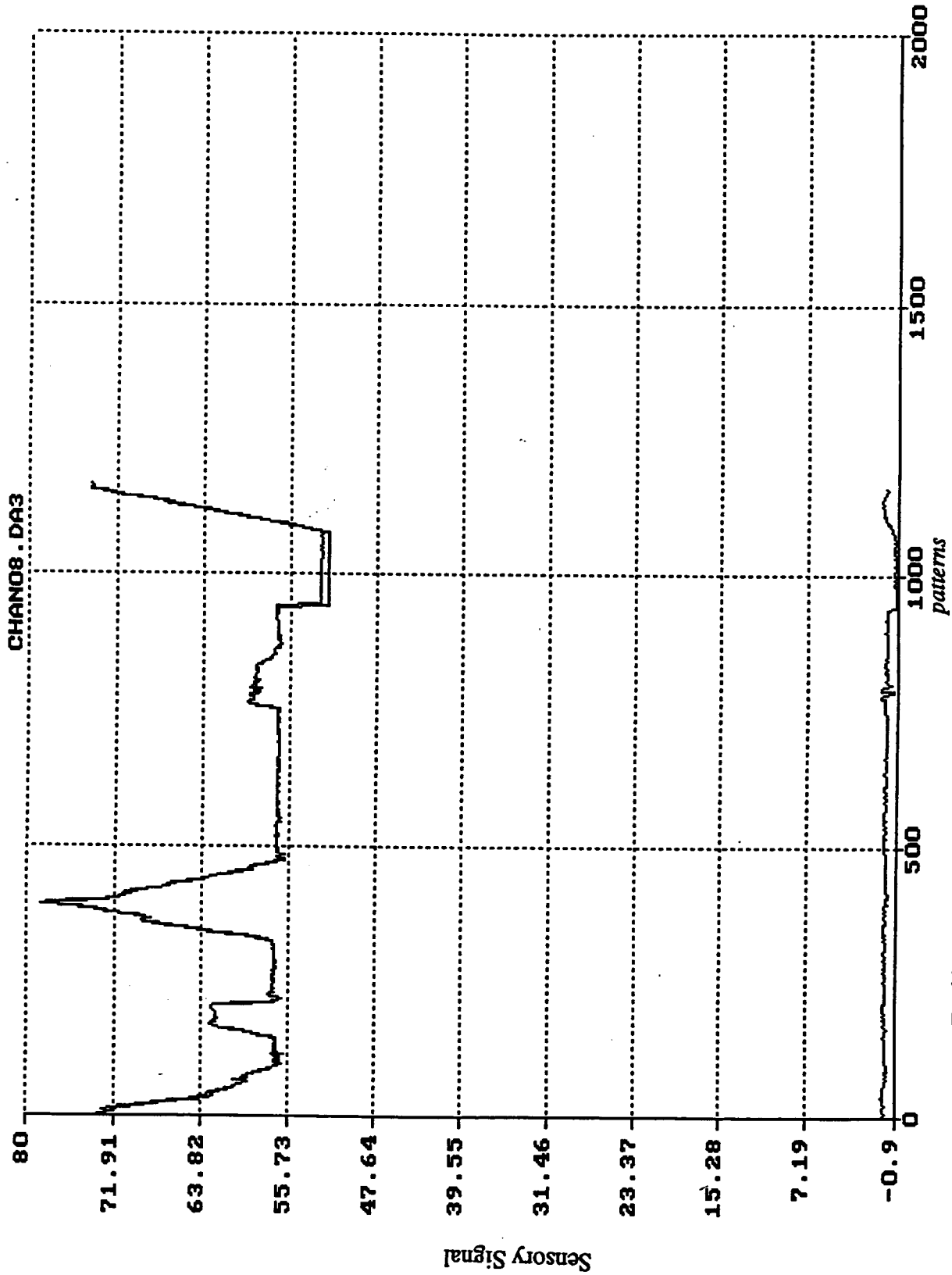

Fig.21: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.
The lowermost graph indicates the error involved. The analysis is carried out for signal #8.

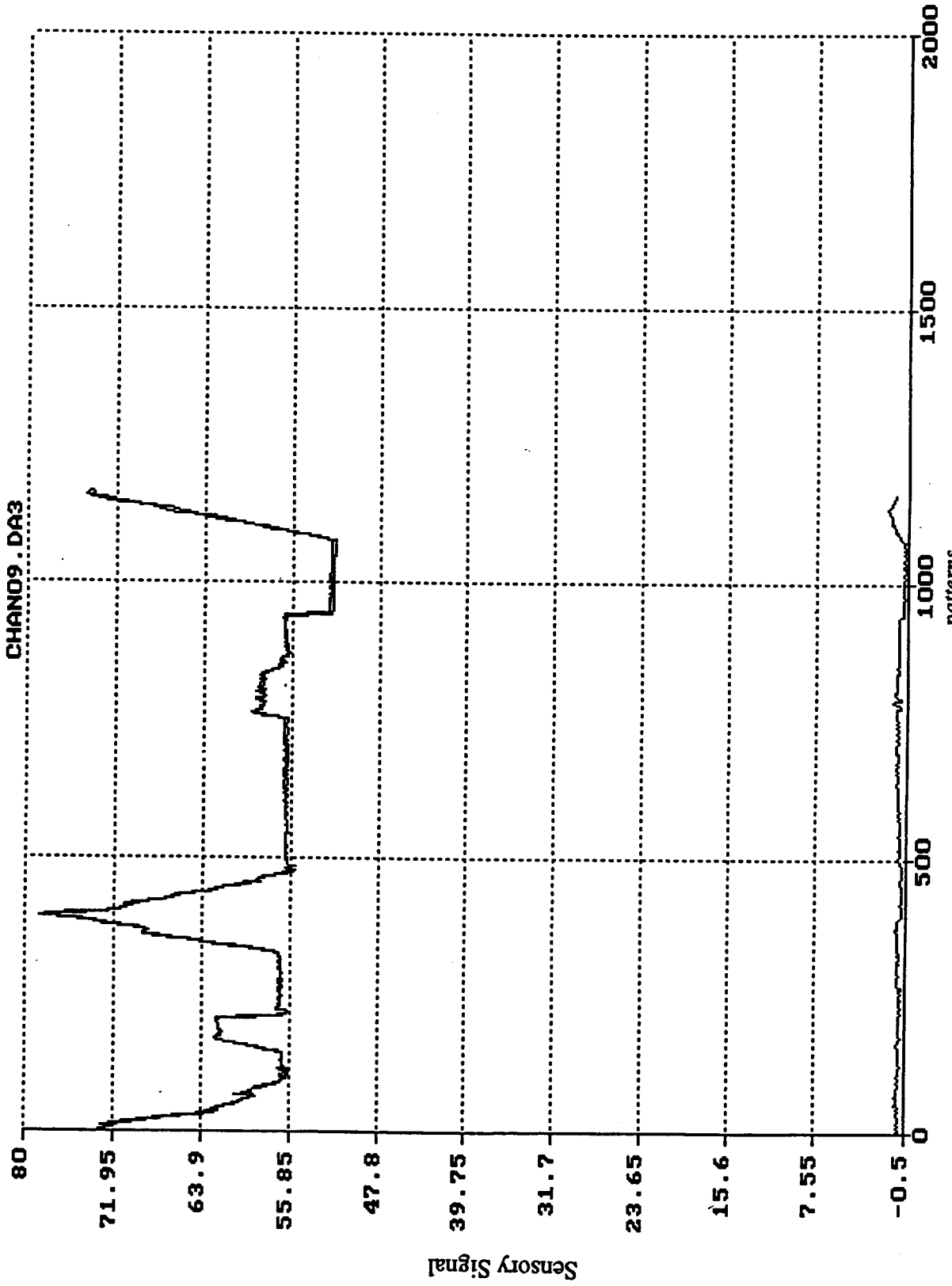

Fig.22: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.
The lowermost graph indicates the error involved. The analysis is carried out for signal #9.

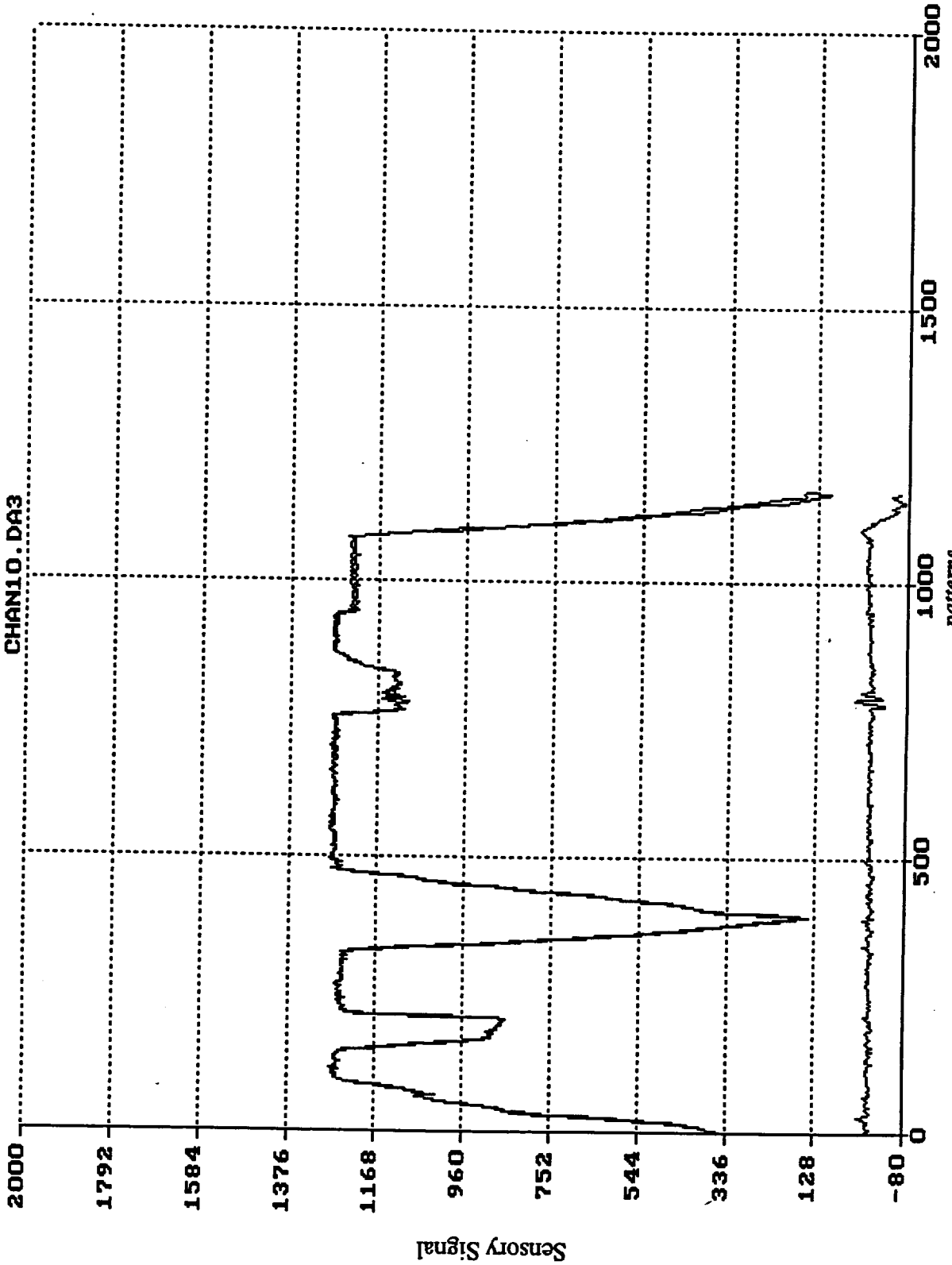

Fig.23: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169. The lowermost graph indicates the error involved. The analysis is carried out for signal #10.

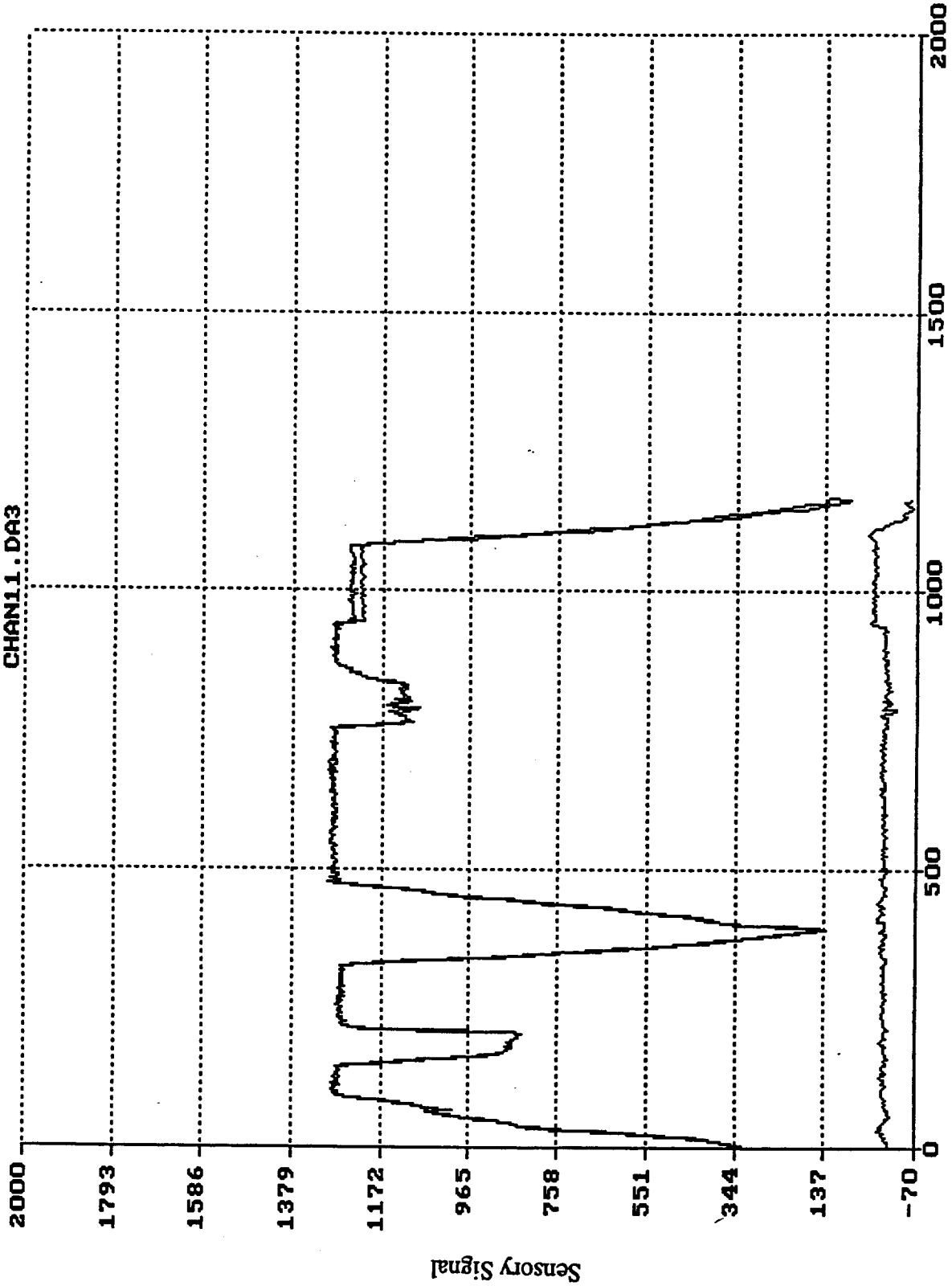

Fig.24: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169.
The lowermost graph indicates the error involved. The analysis is carried out for signal #11.

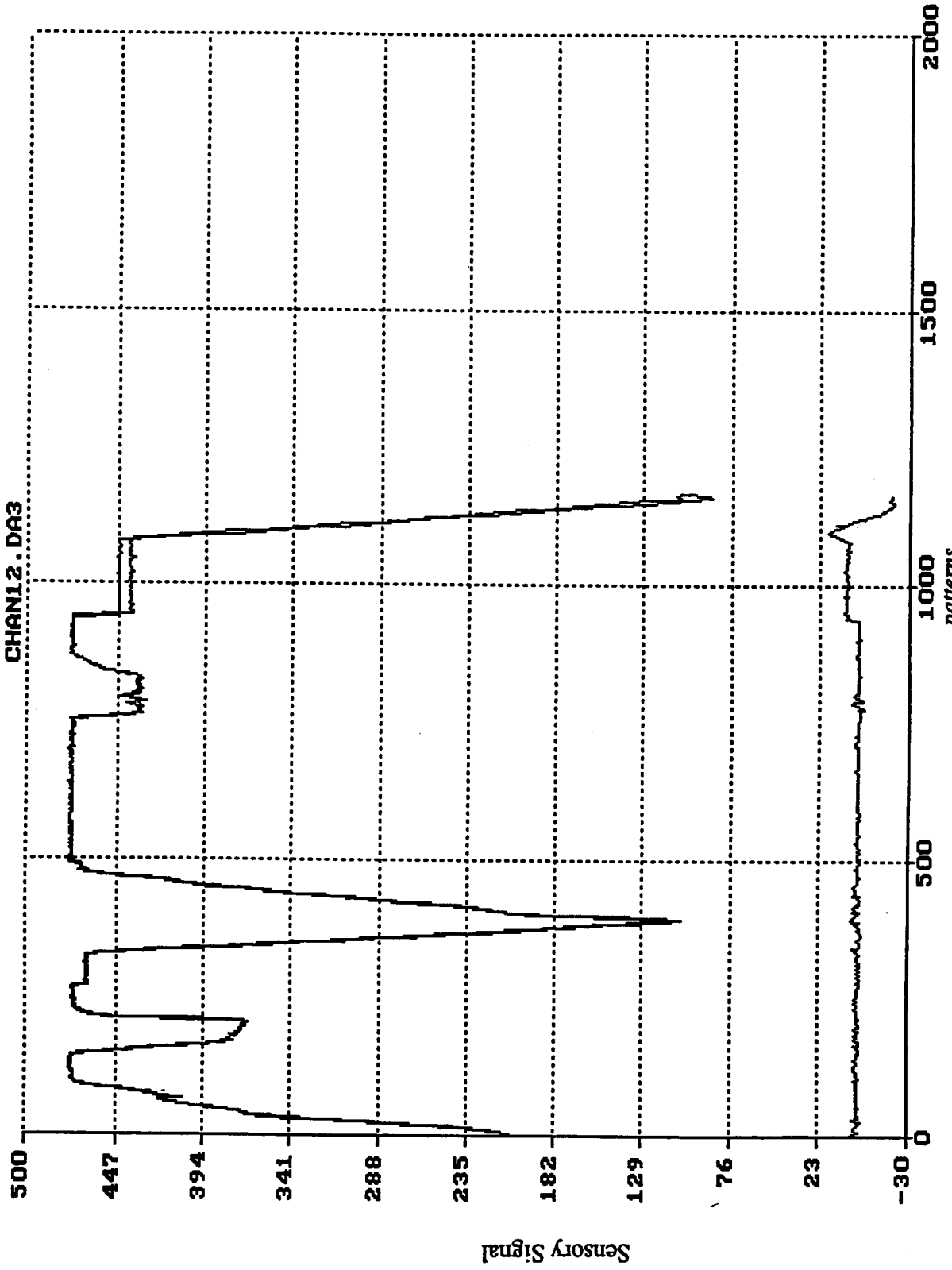

Fig.25: Processing the benchmark data. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total number of patterns which is 1169. The lowermost graph indicates the error involved. The analysis is carried out for signal #12

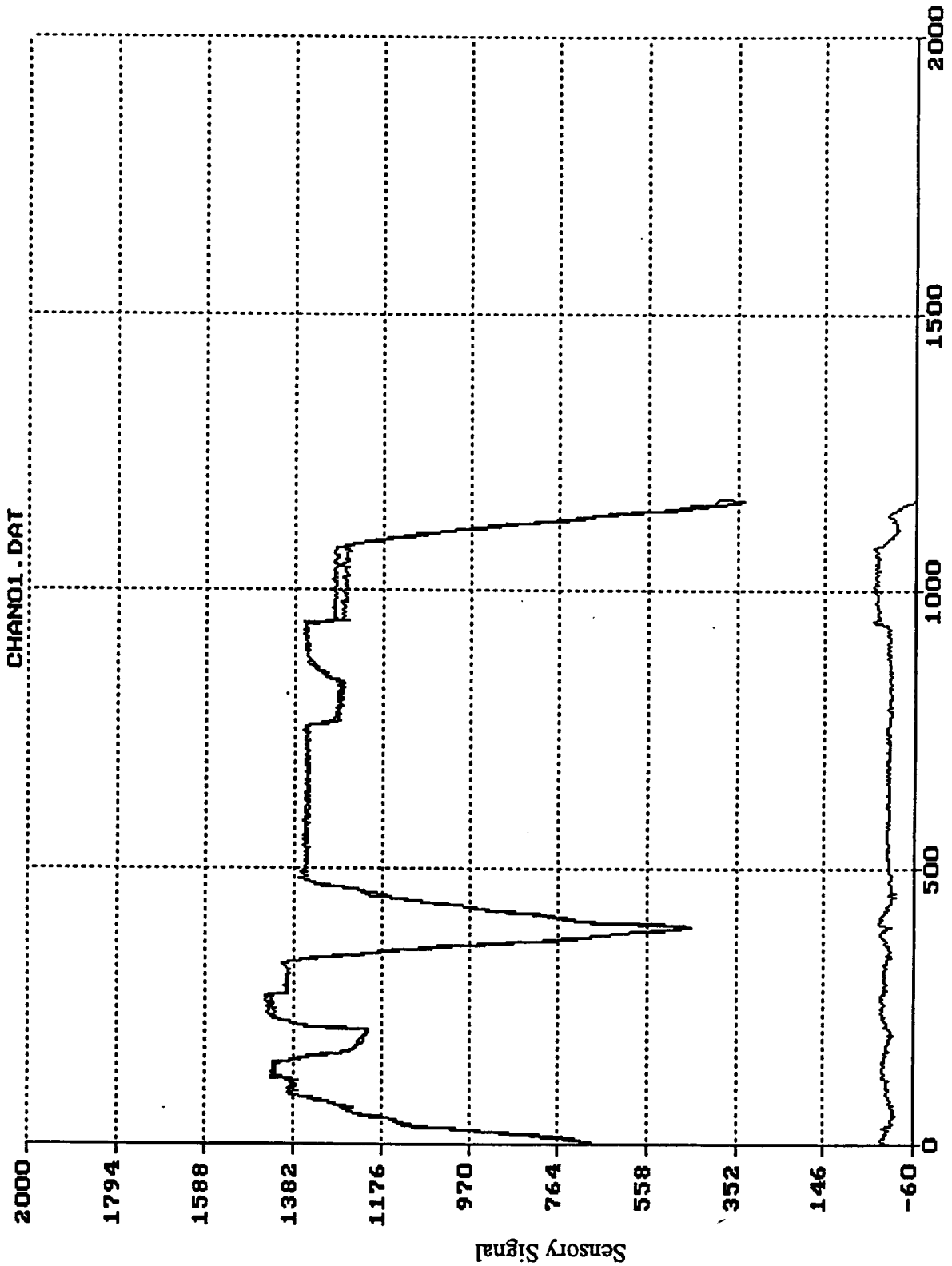

Fig.26: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #1

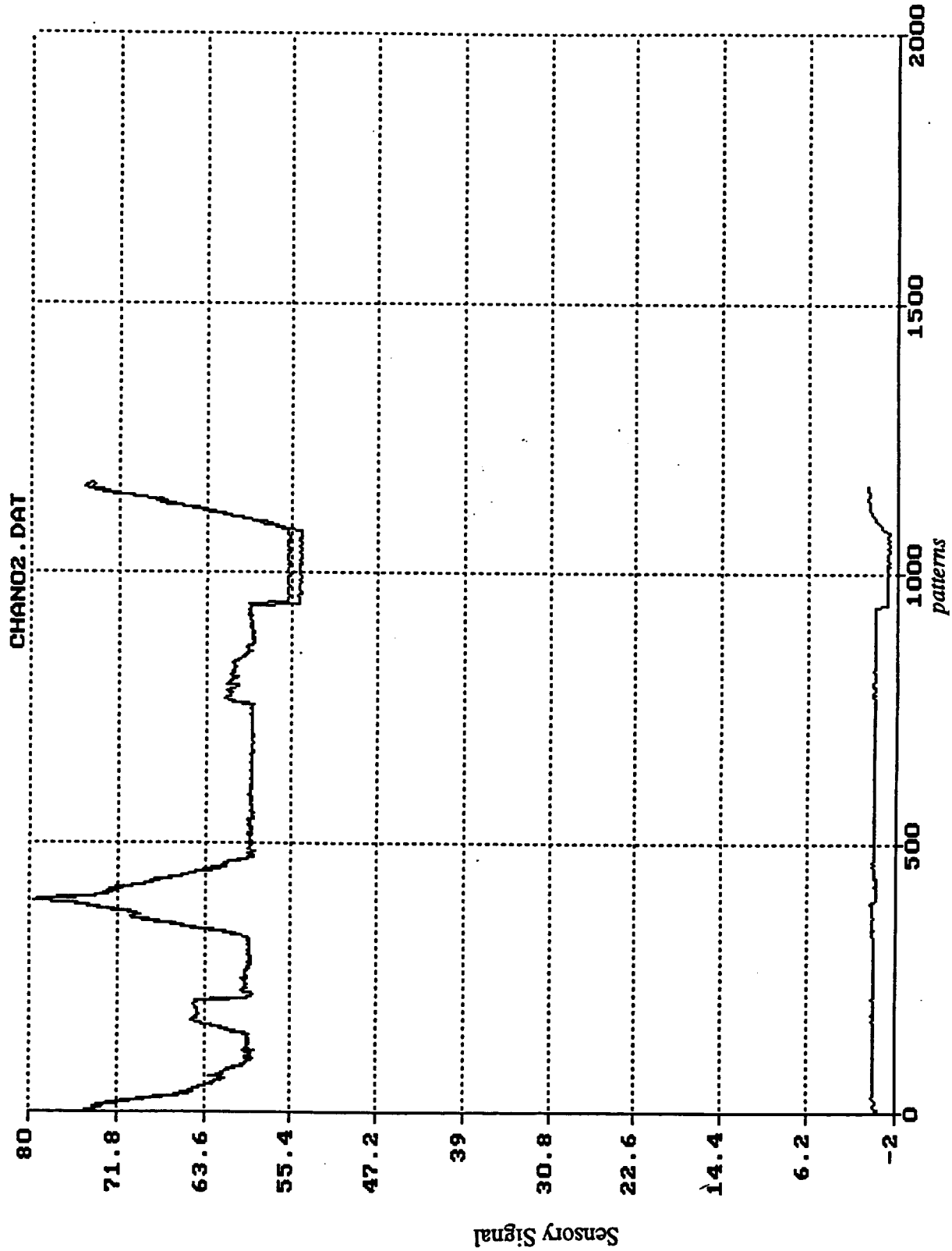

Fig.27: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #2.

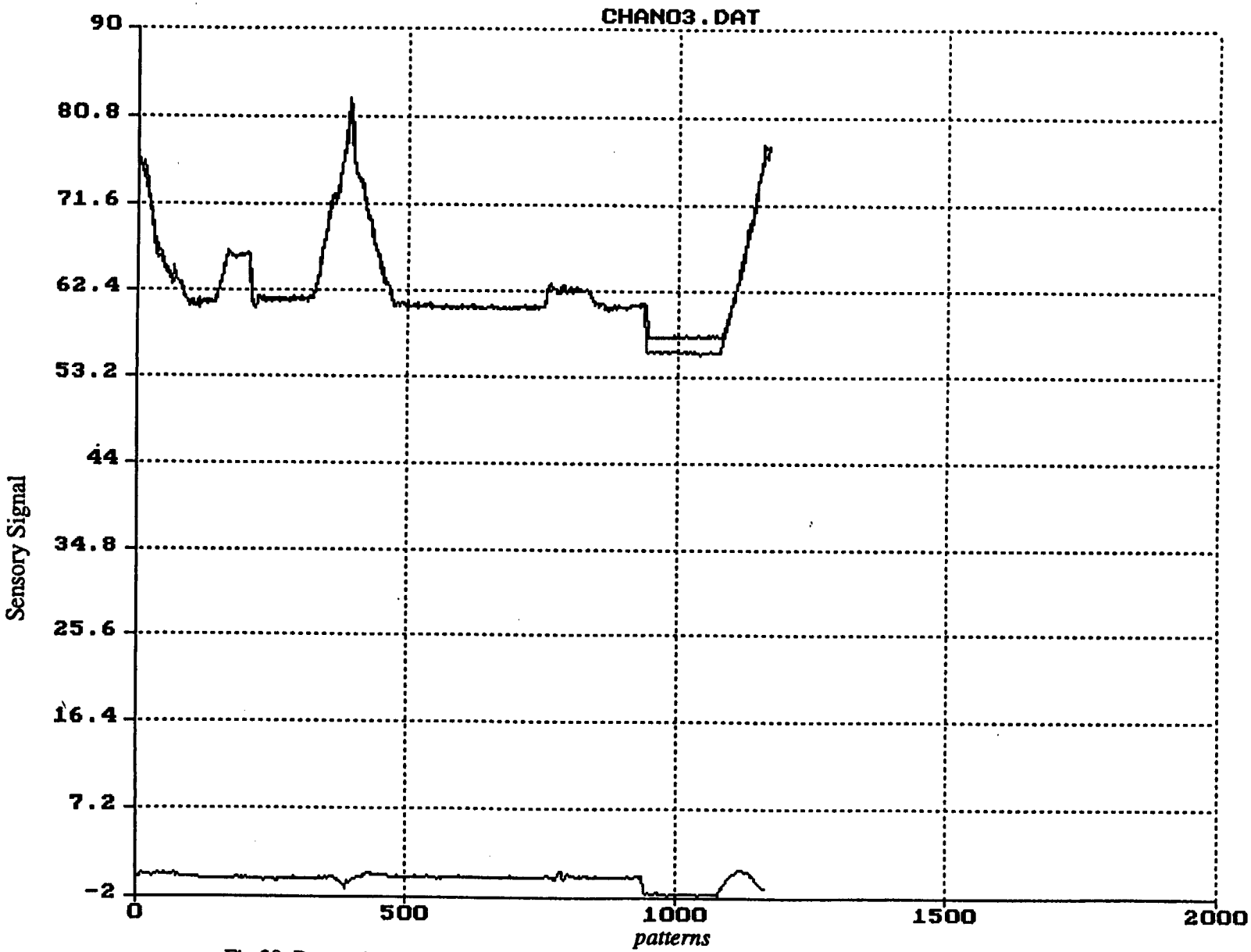

Fig.28: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #3.

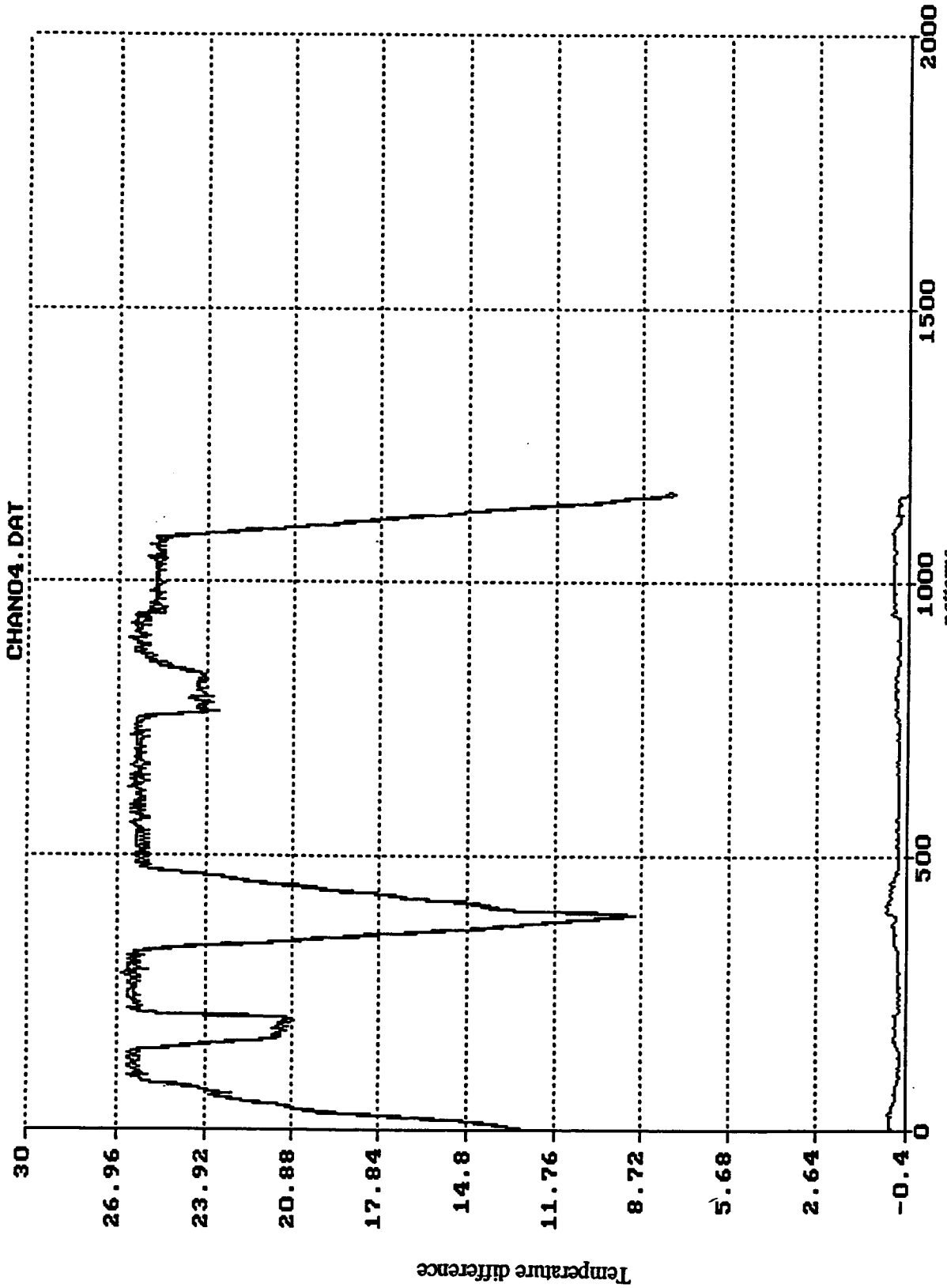

Fig.29: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #4.

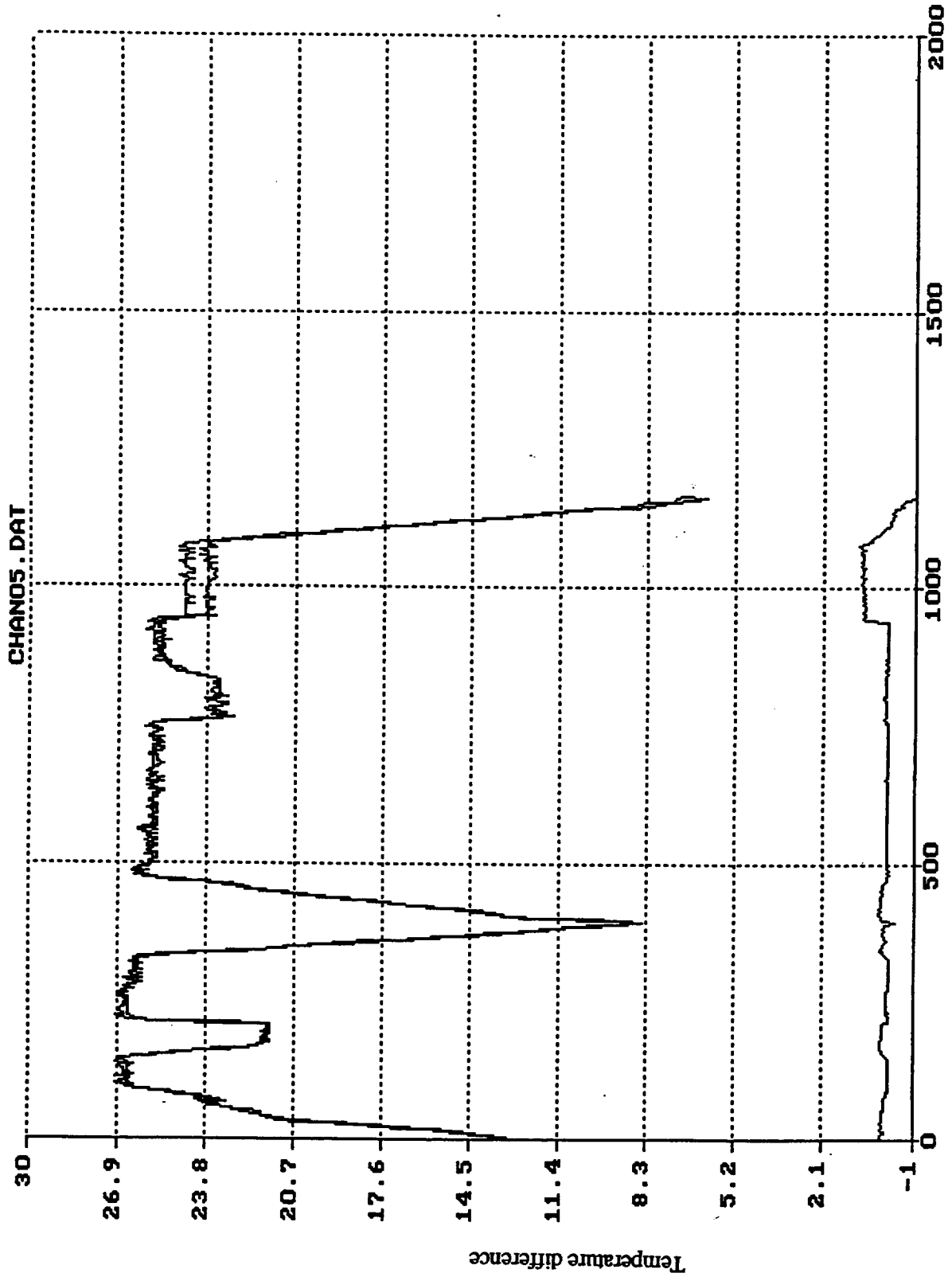

Fig.30: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #5.

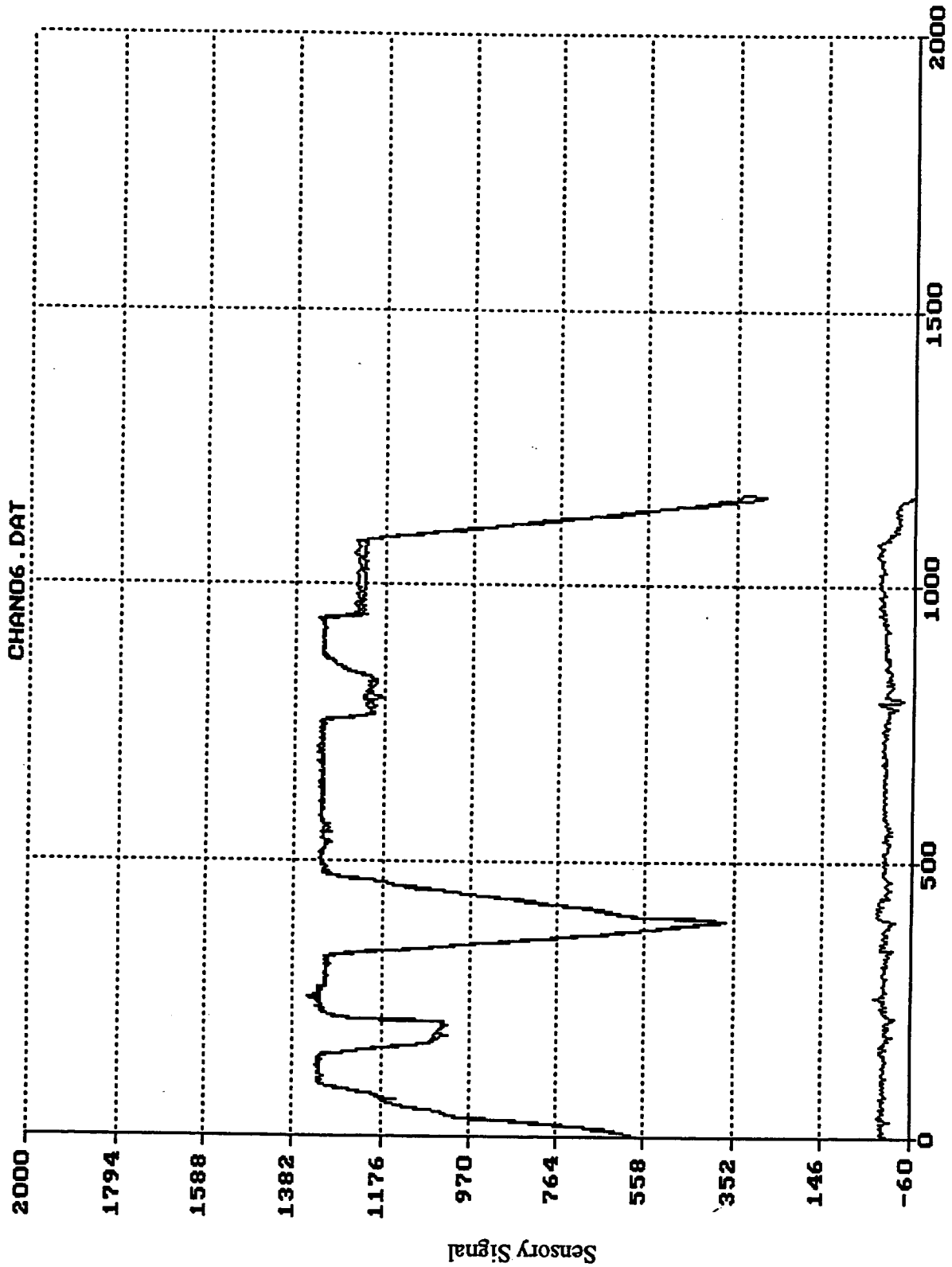

Fig.31: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #6.

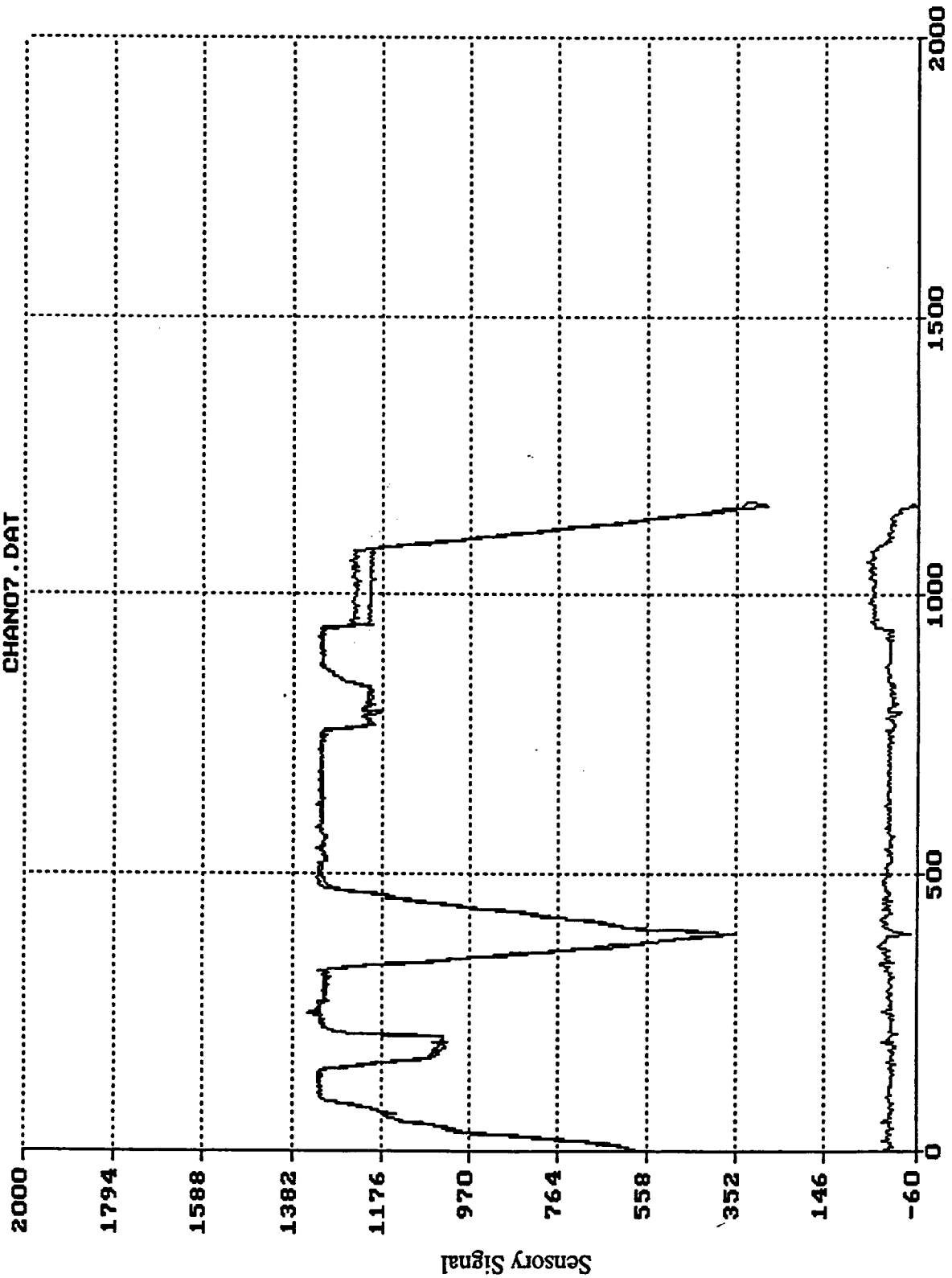

Fig.32: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #7.

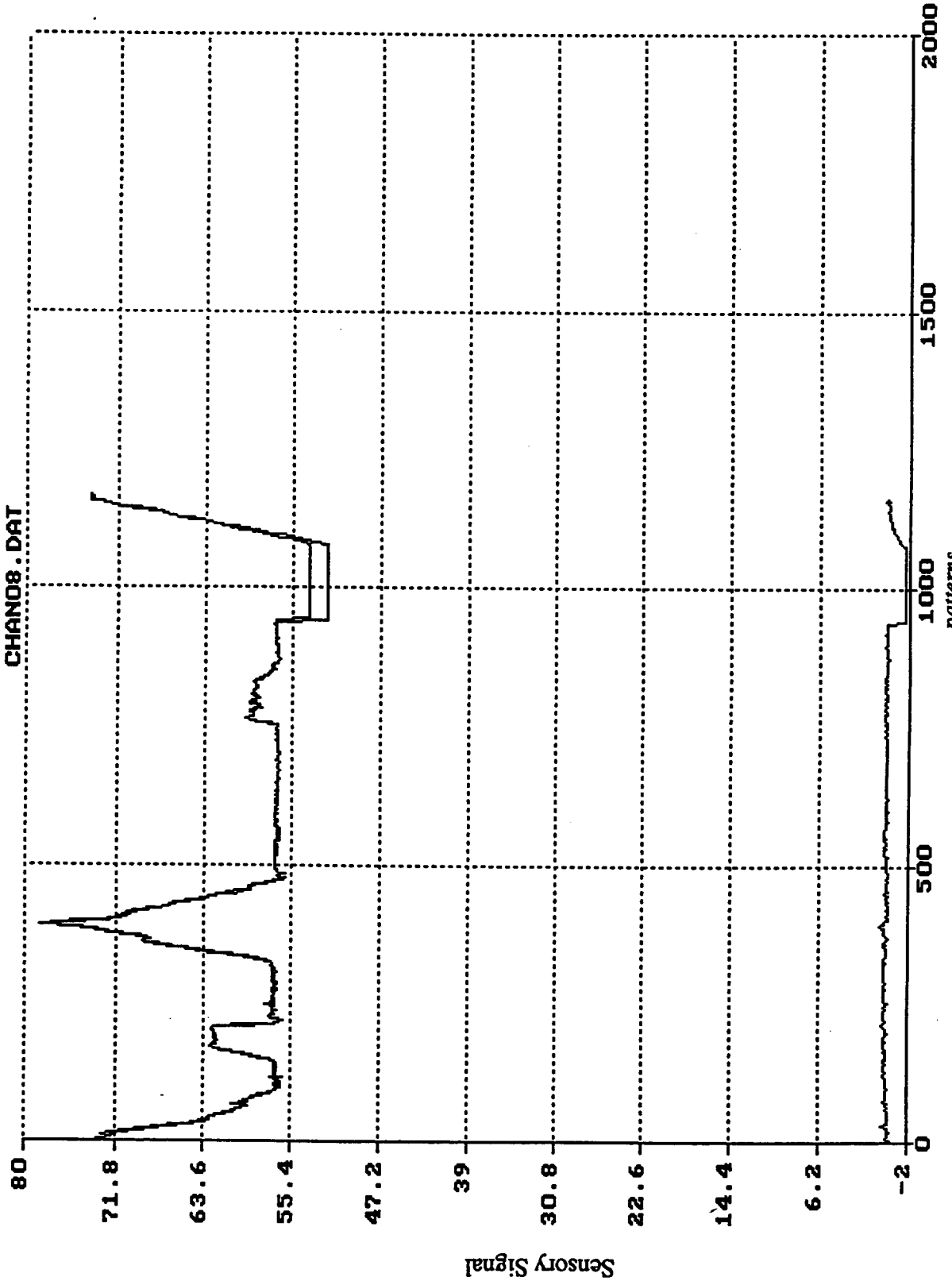

Fig.33: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #8.

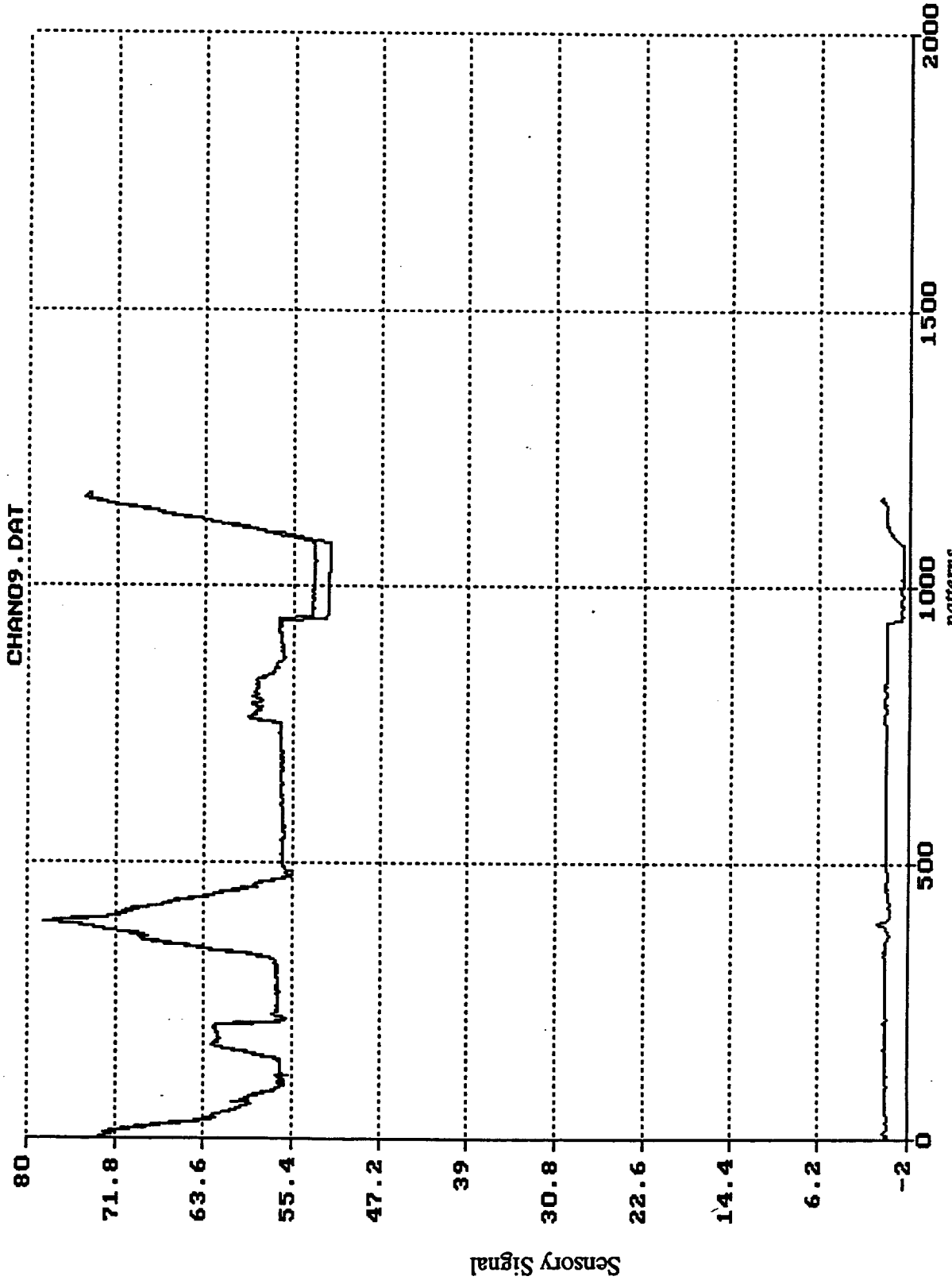

Fig.34: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #9.

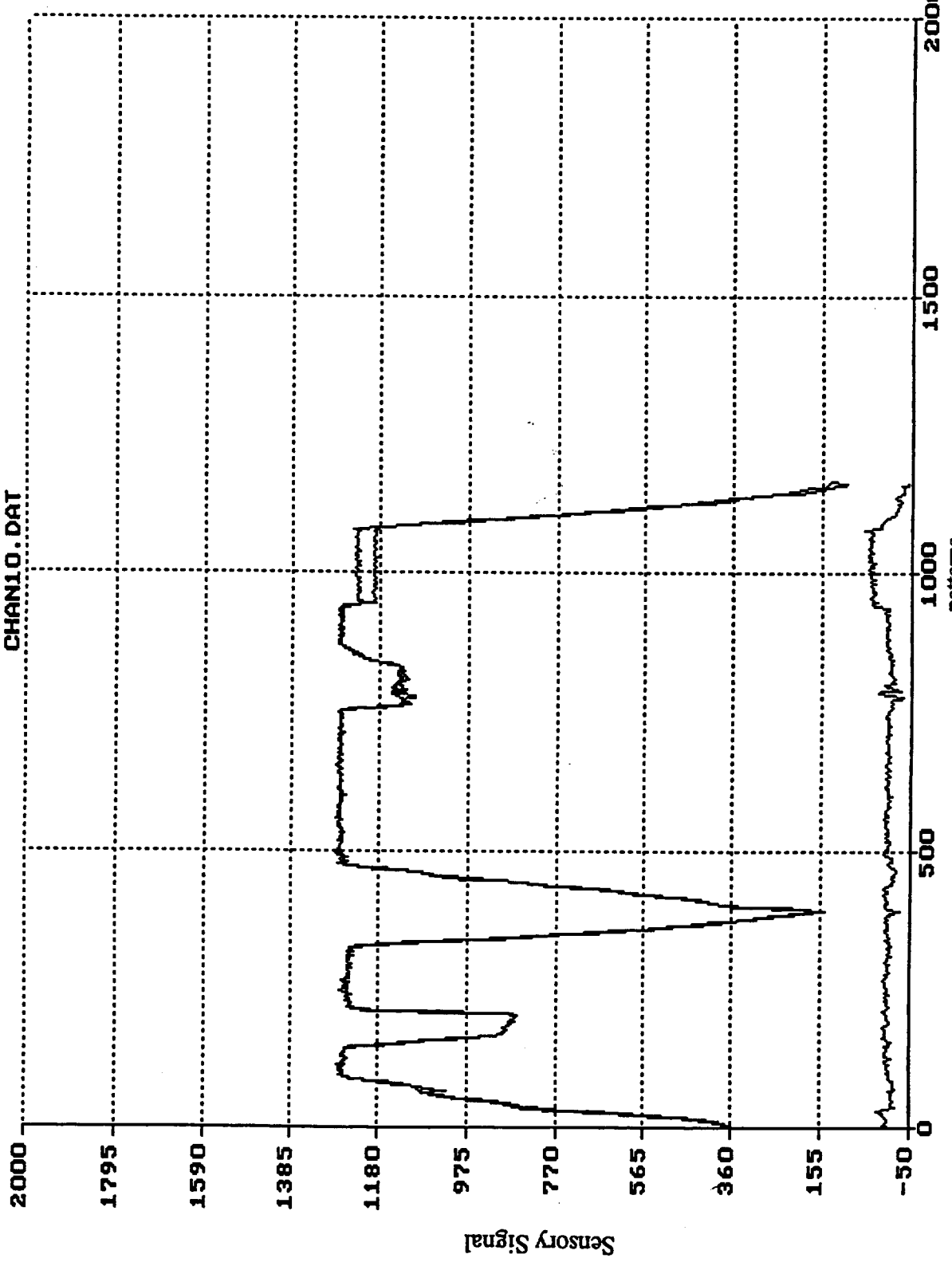

Fig.35: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #10.

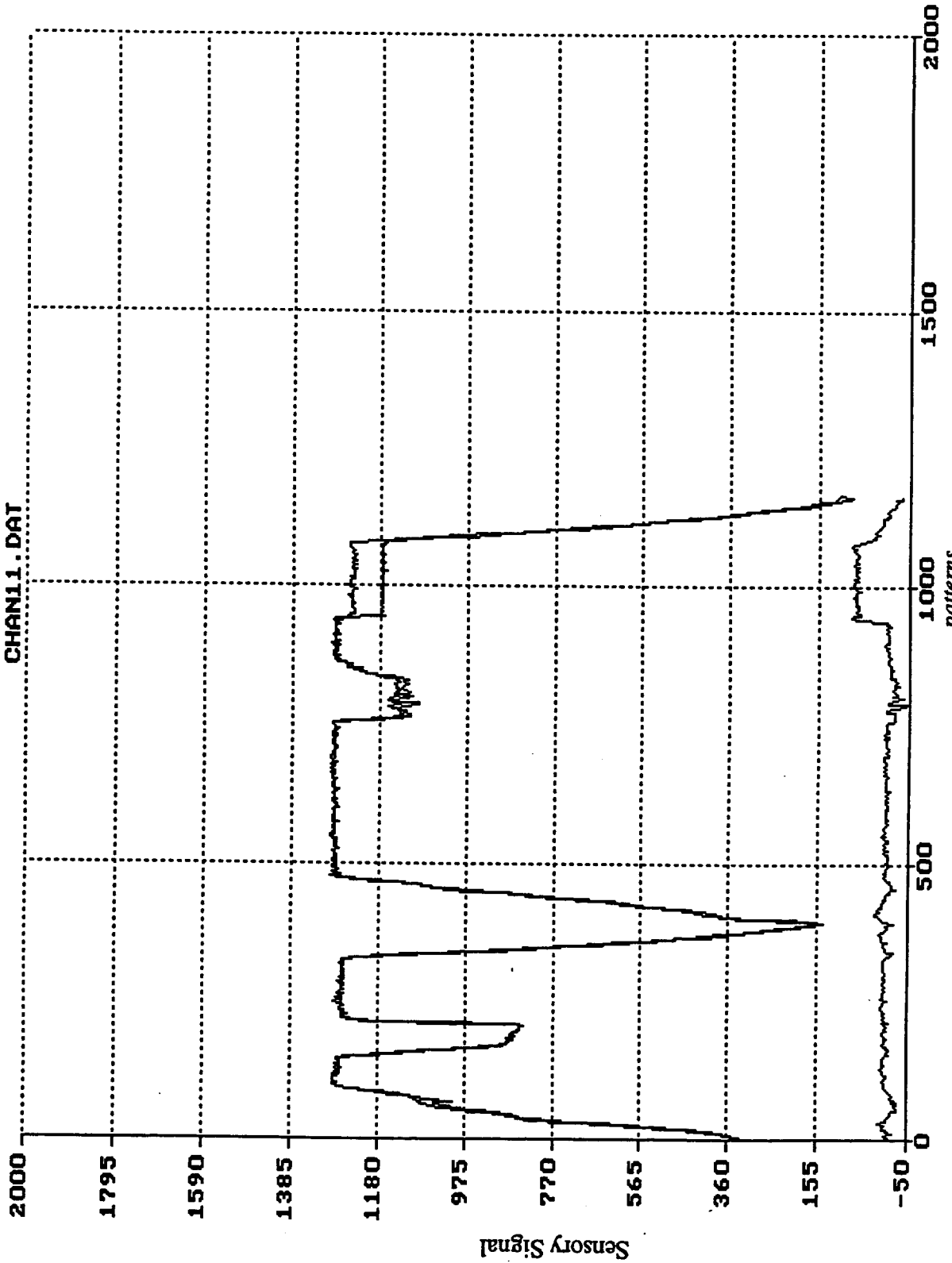

Fig.36: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #11.

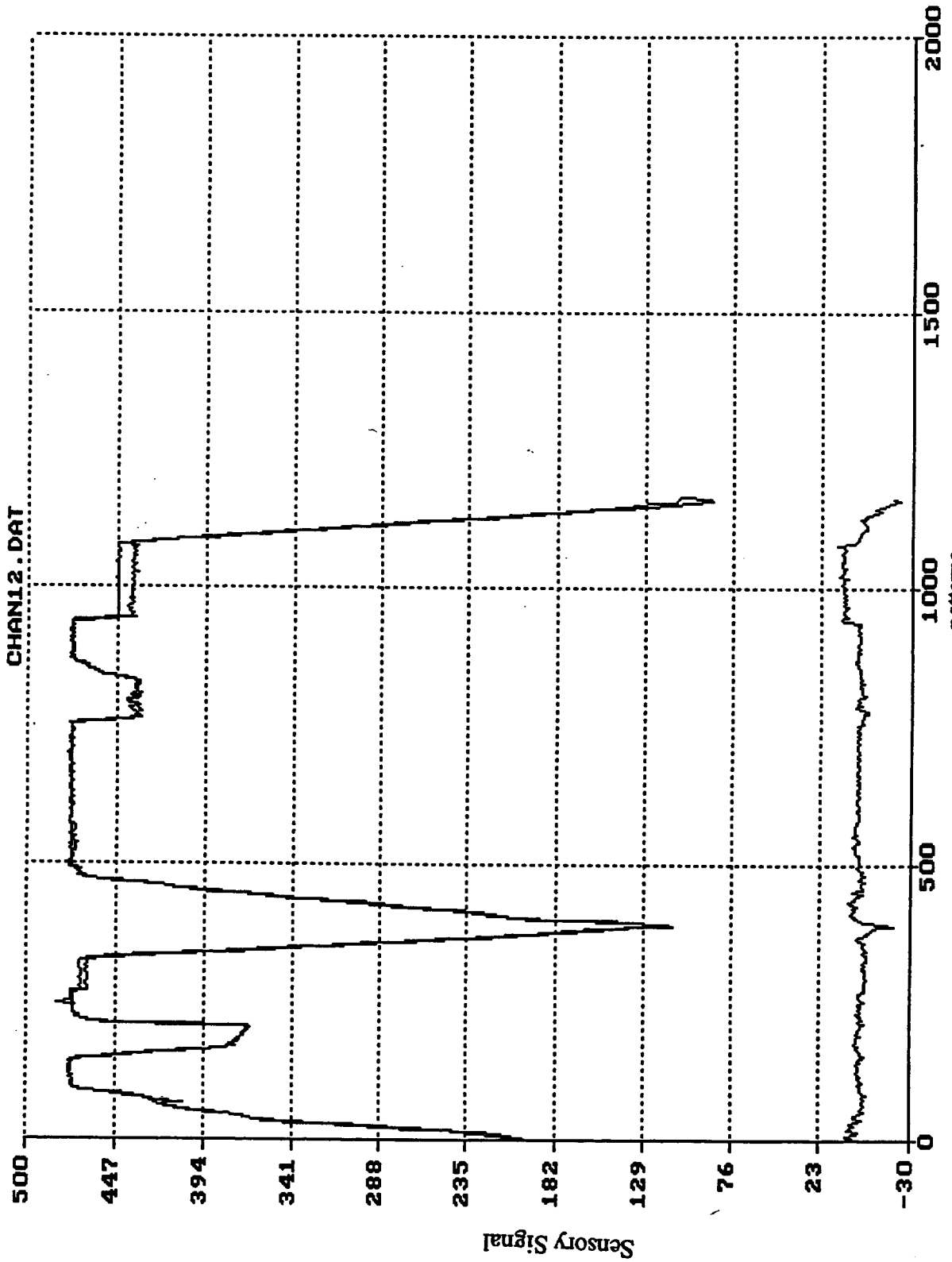

Fig.37: Processing the benchmark data with additive noise. The network structure is autoassociative and the training method is variable-metric minimization. For training 600 patterns from the beginning are used. The recall is performed over the total 1169 patterns. The lowermost graph is the error involved. The analysis is carried out for signal #12.

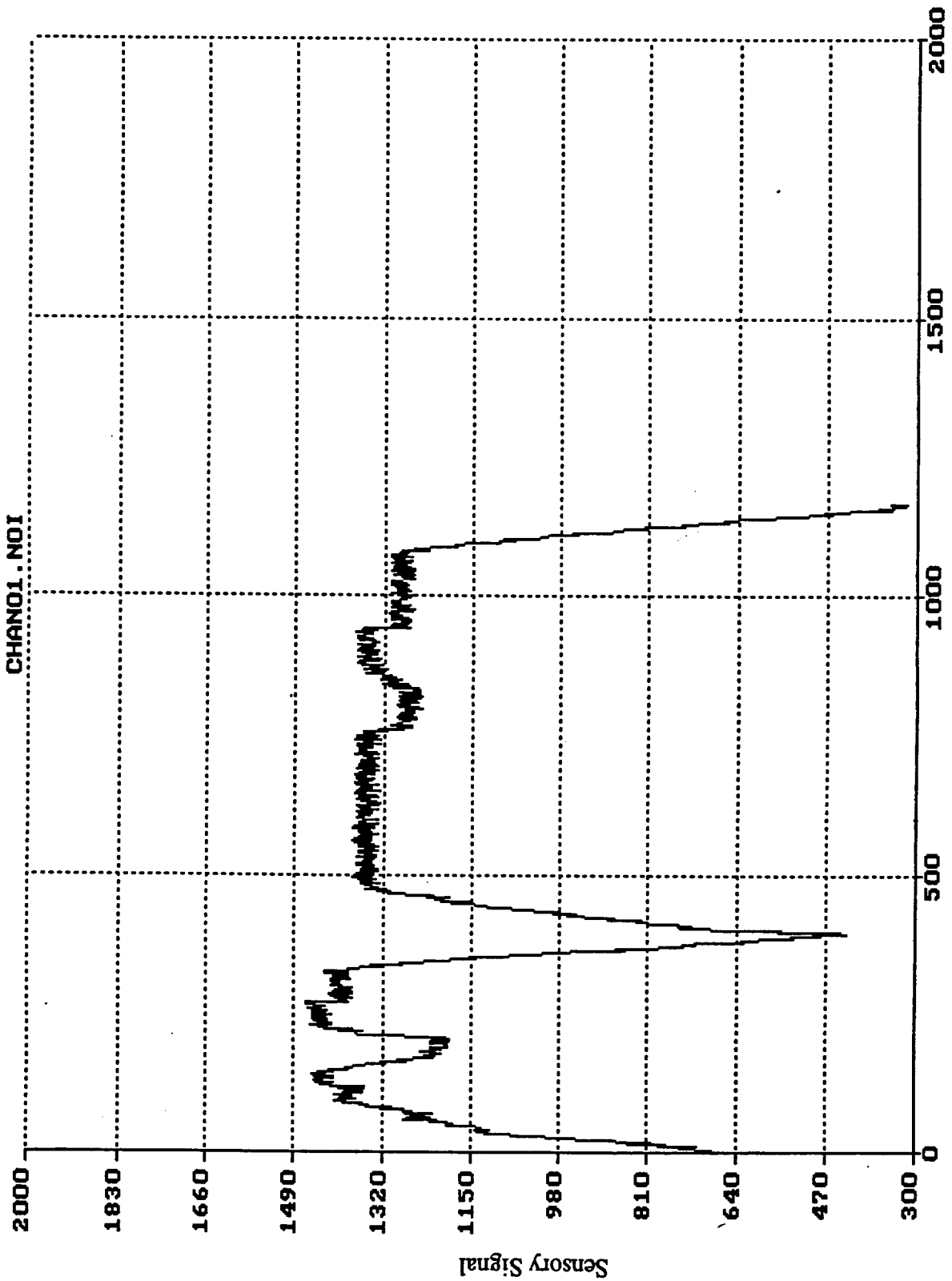

Fig.38: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #1.

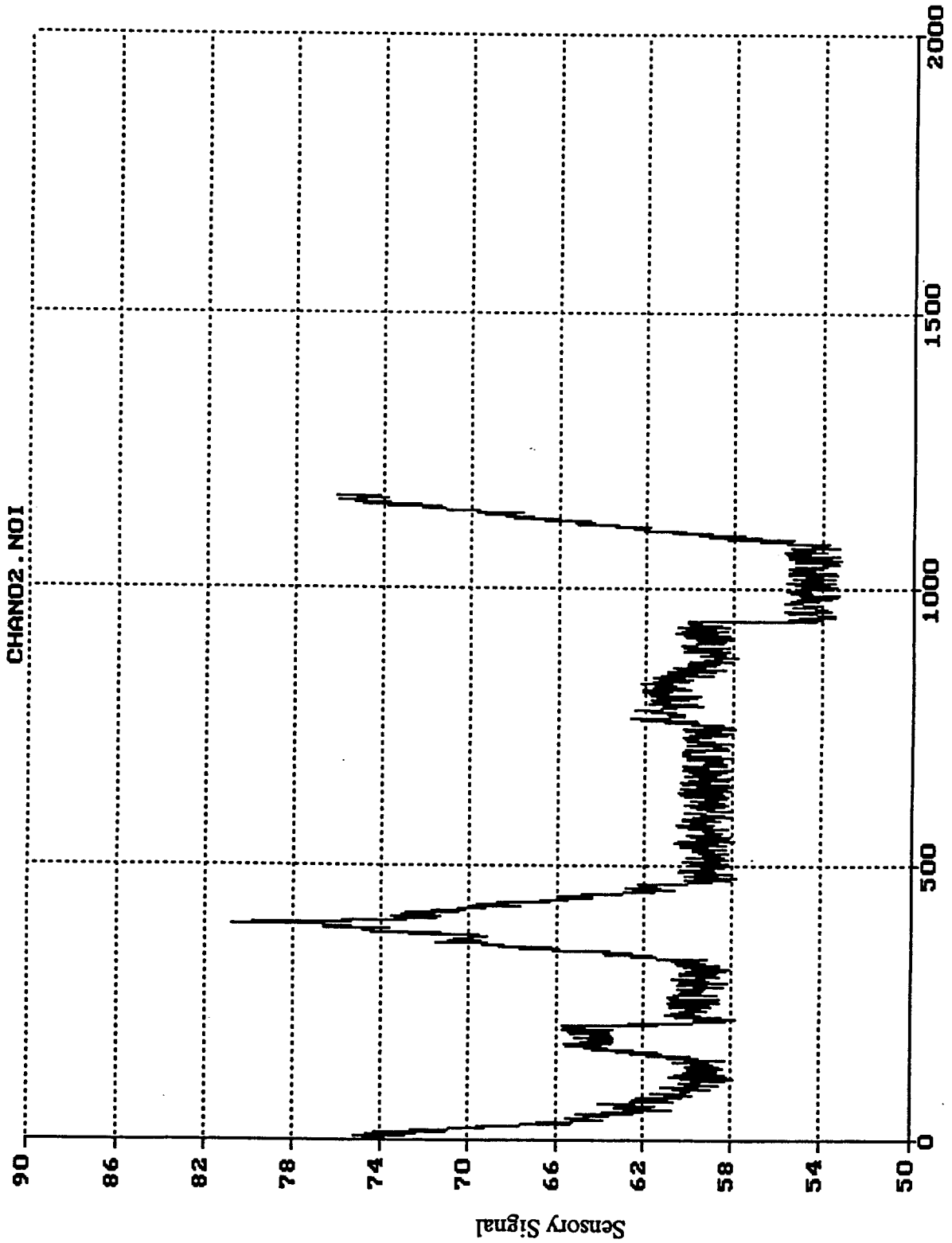

Fig.39: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #2.

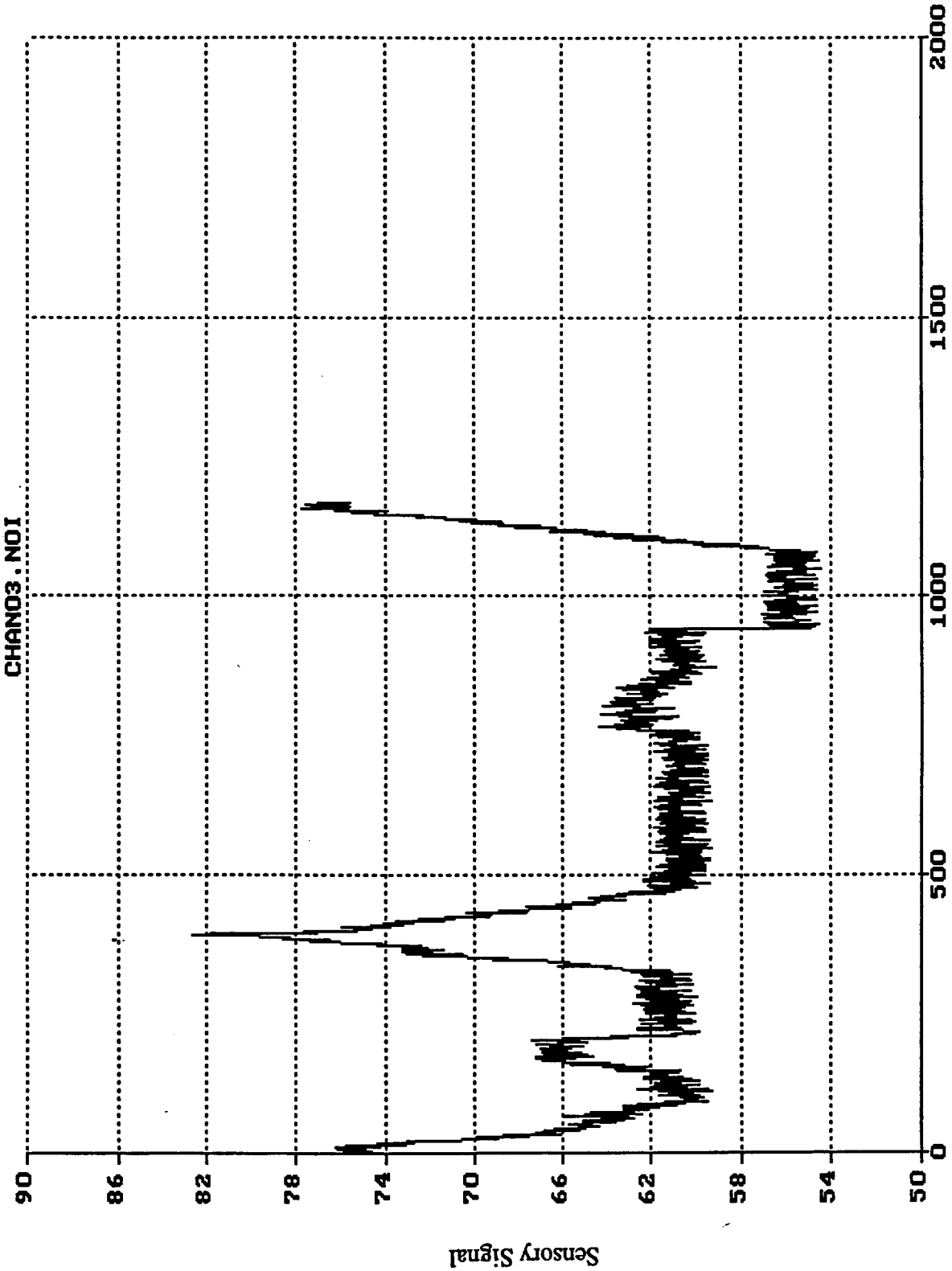

Fig.40: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #3.

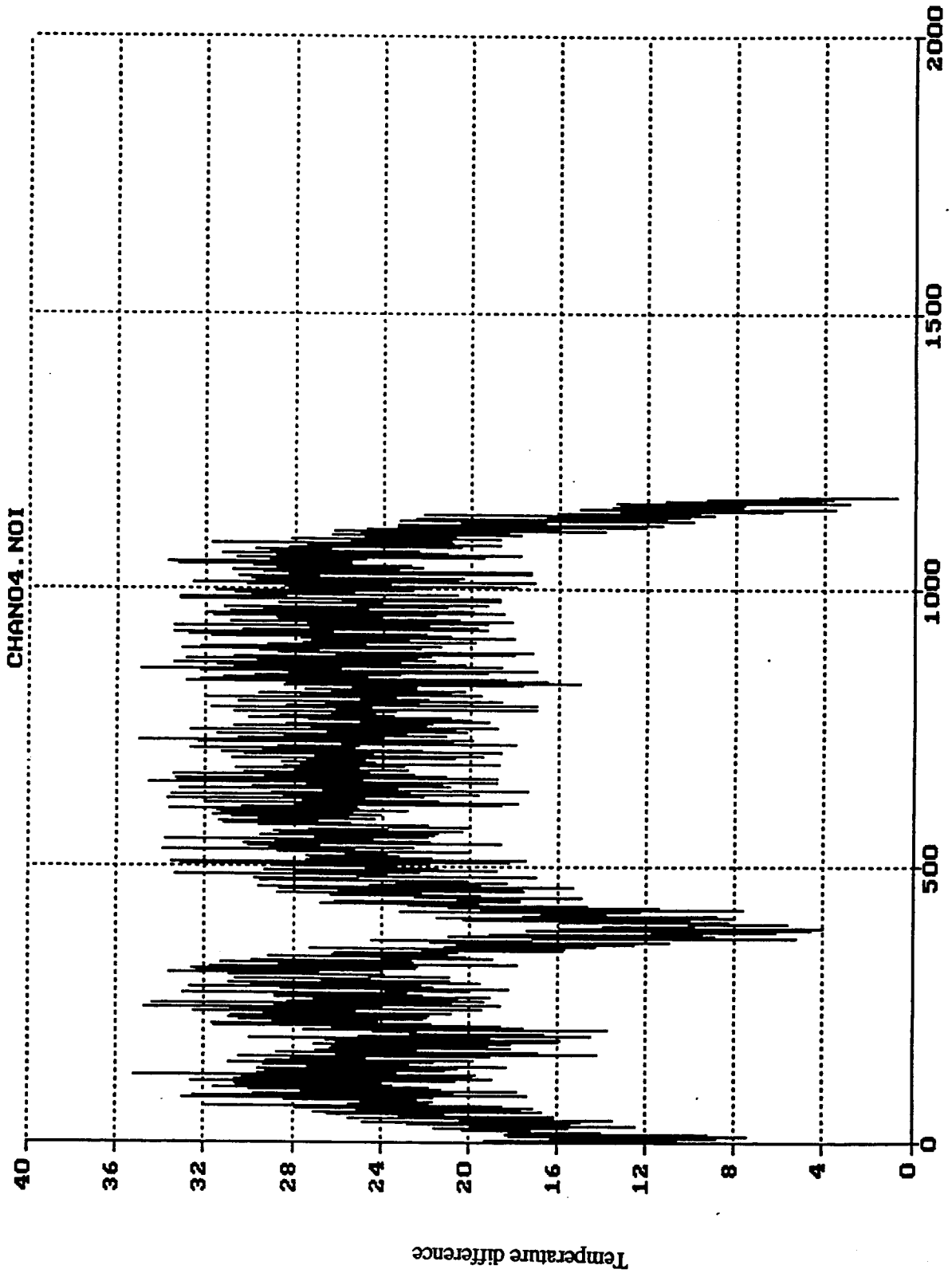

Fig.41: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #4.

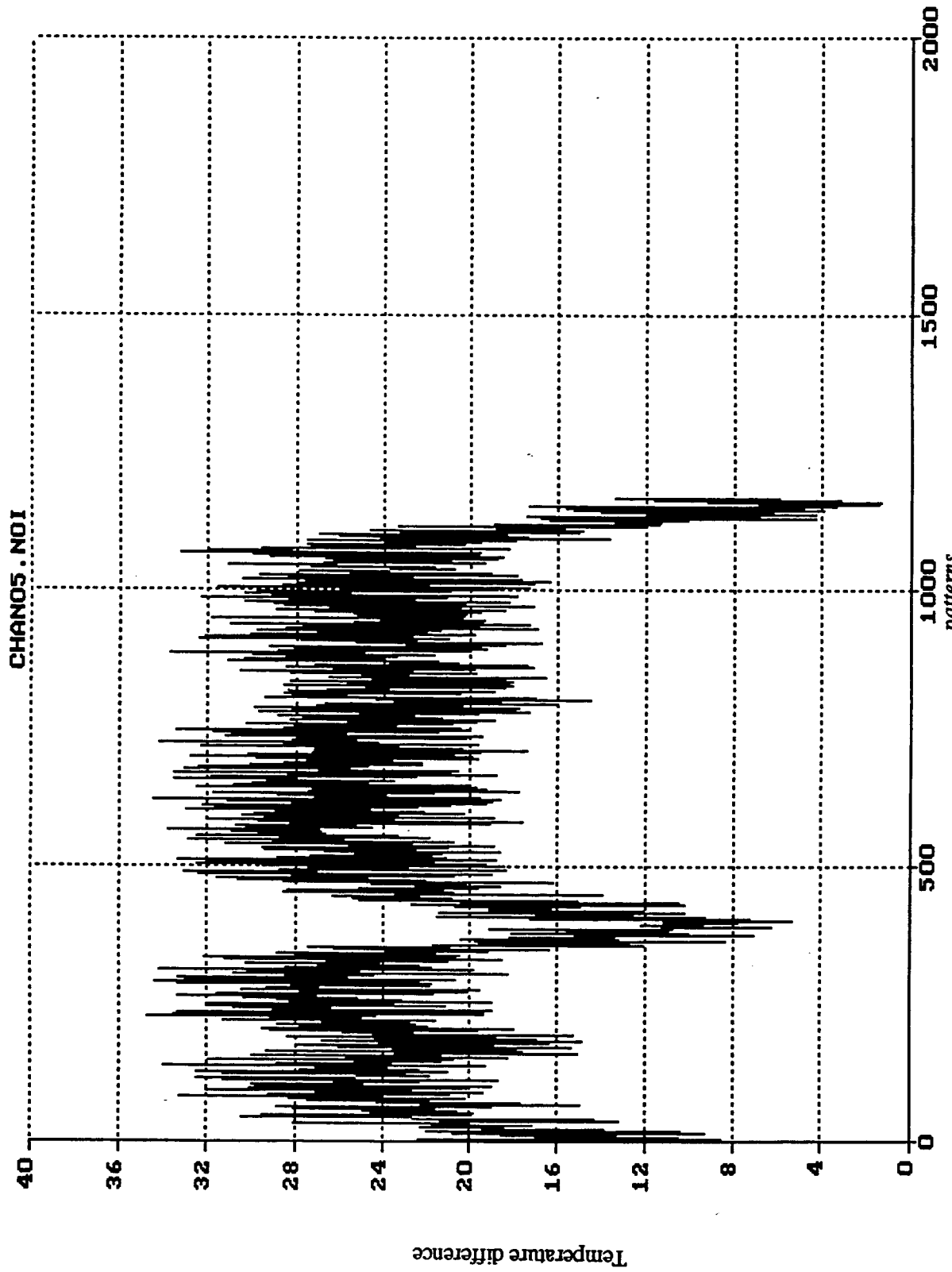

Fig.42: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #5.

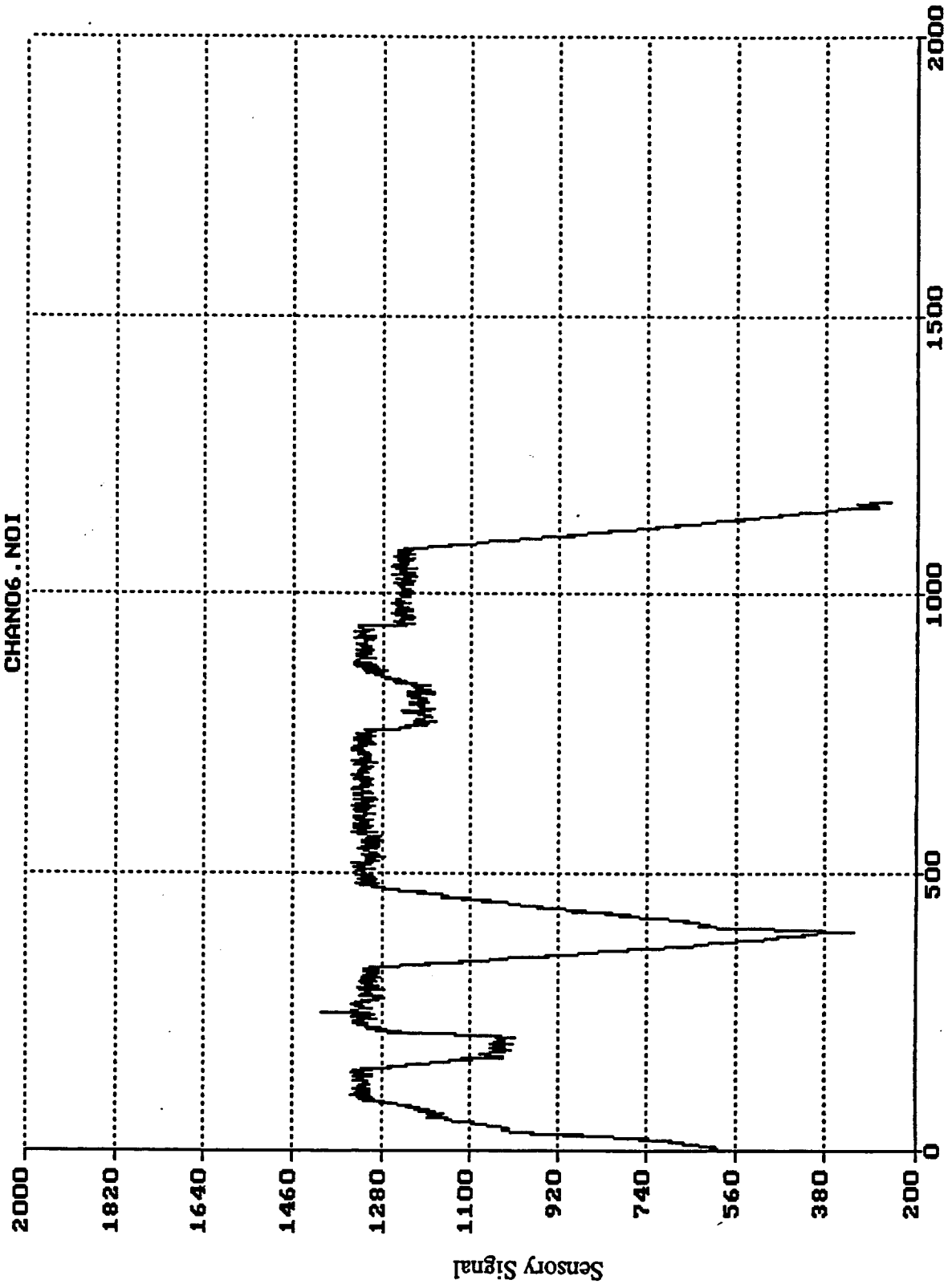

Fig.43: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is fof signal #6.

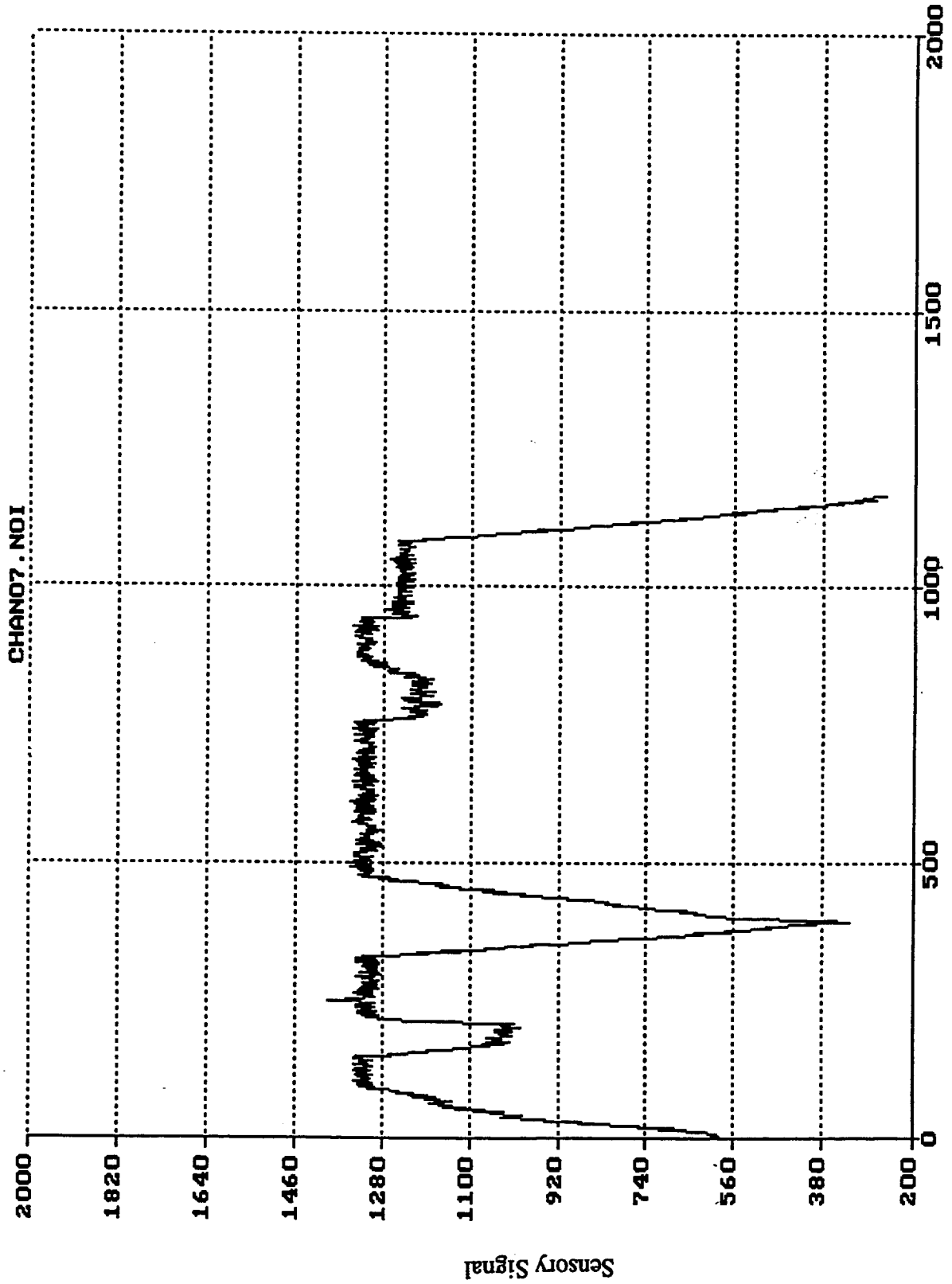

Fig.44: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #7.

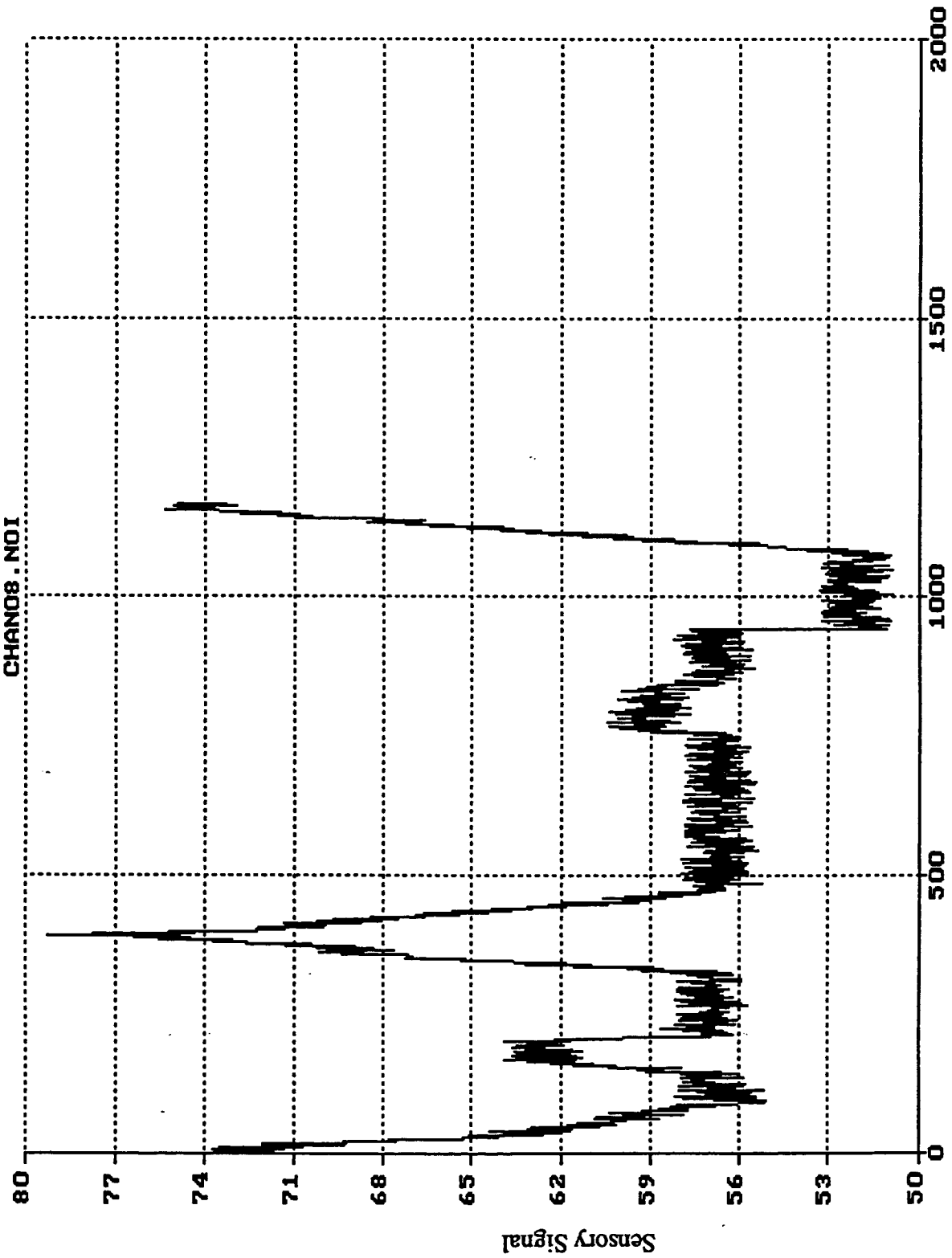

Fig.45: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #8.

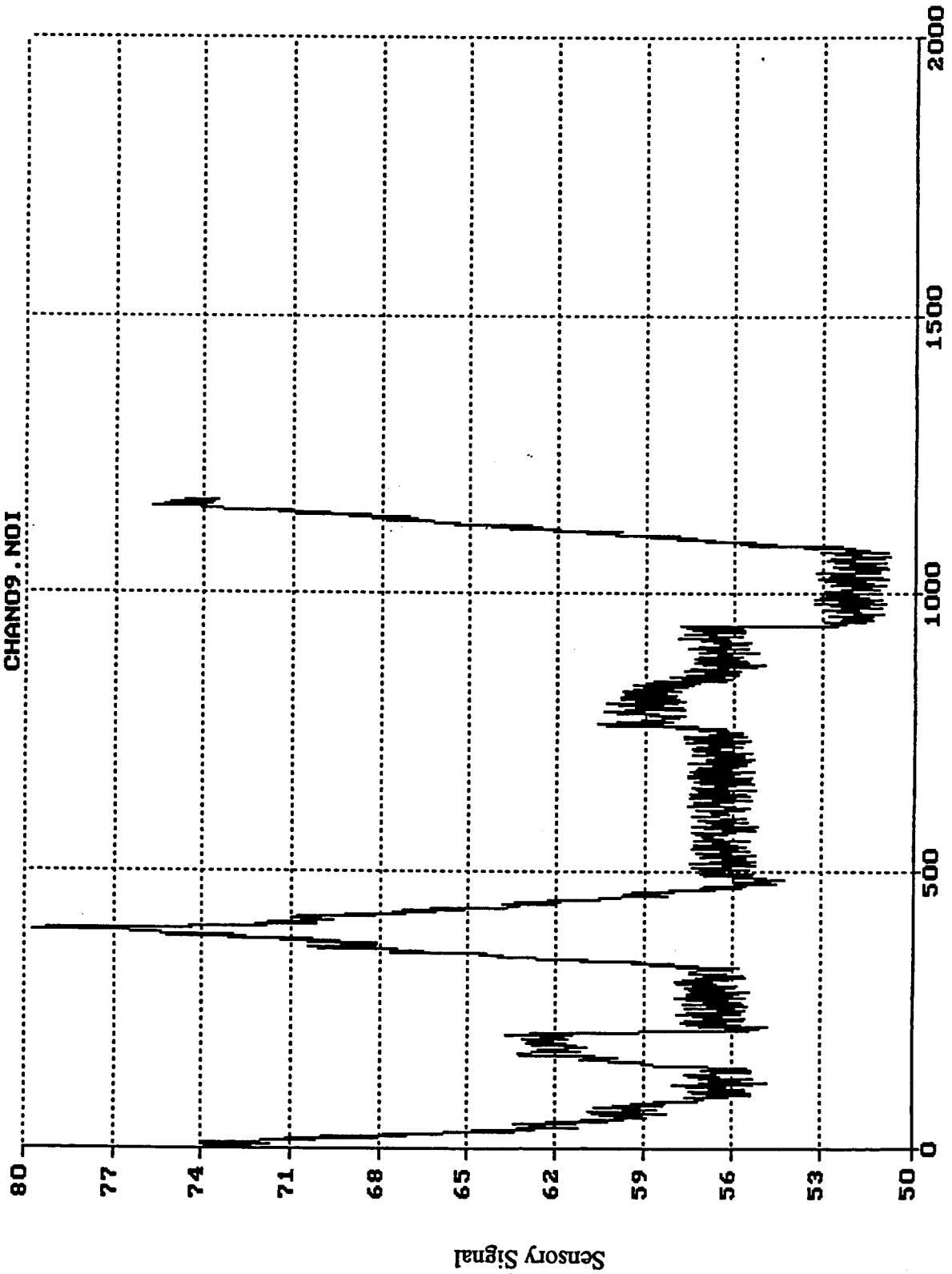

Fig.47: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #9.

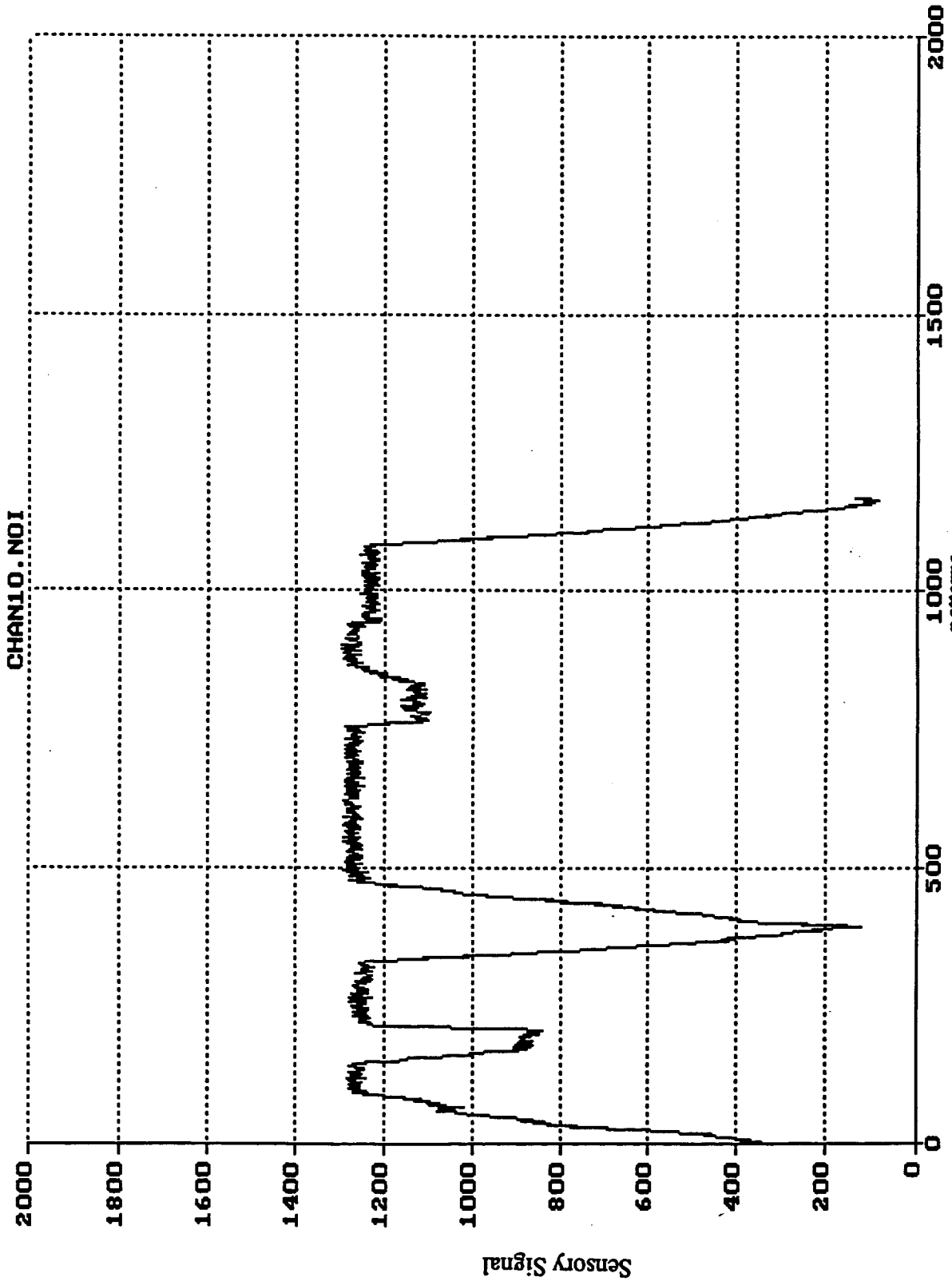

Fig.46: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #10.

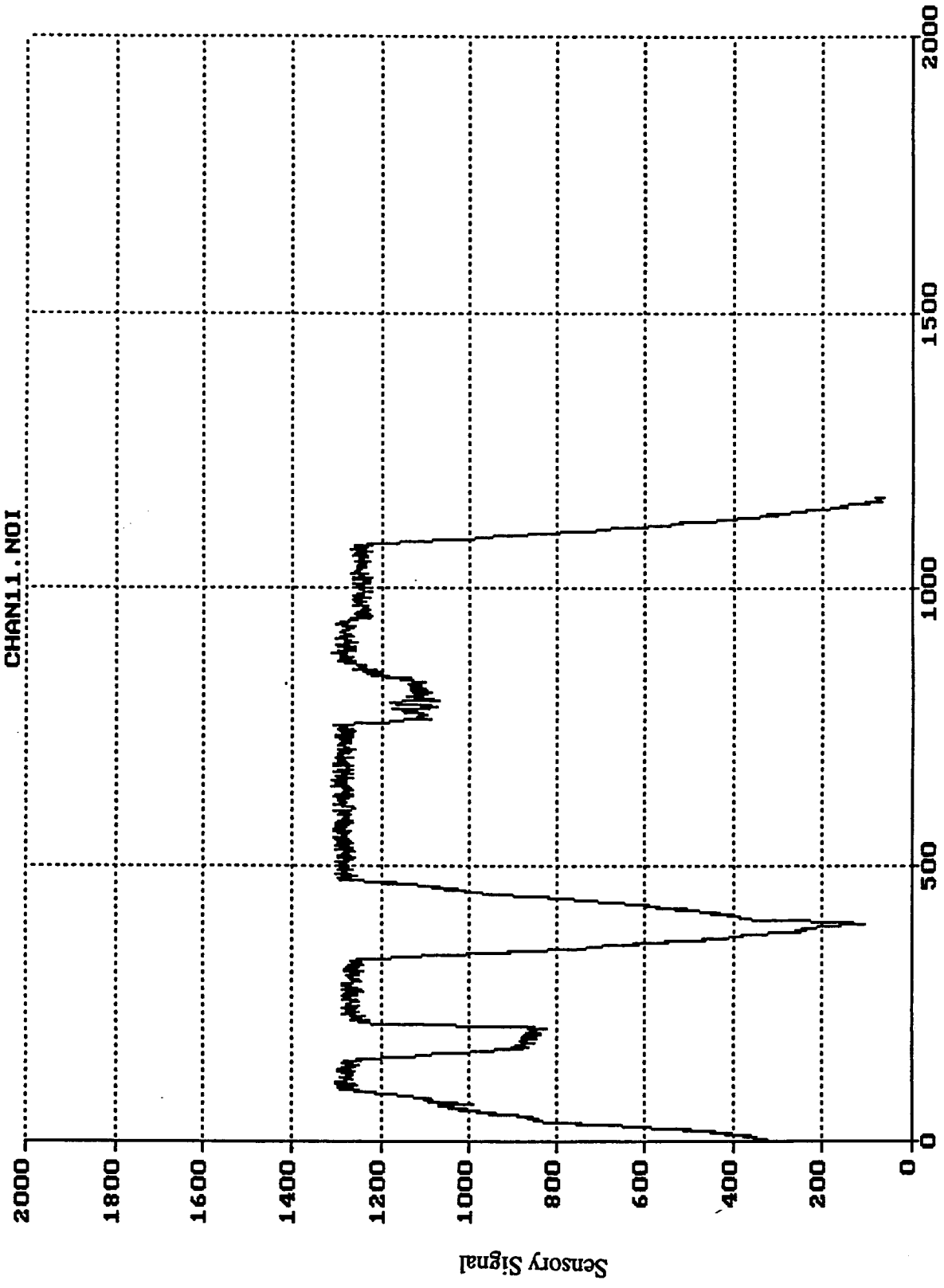

Fig.48: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #11.

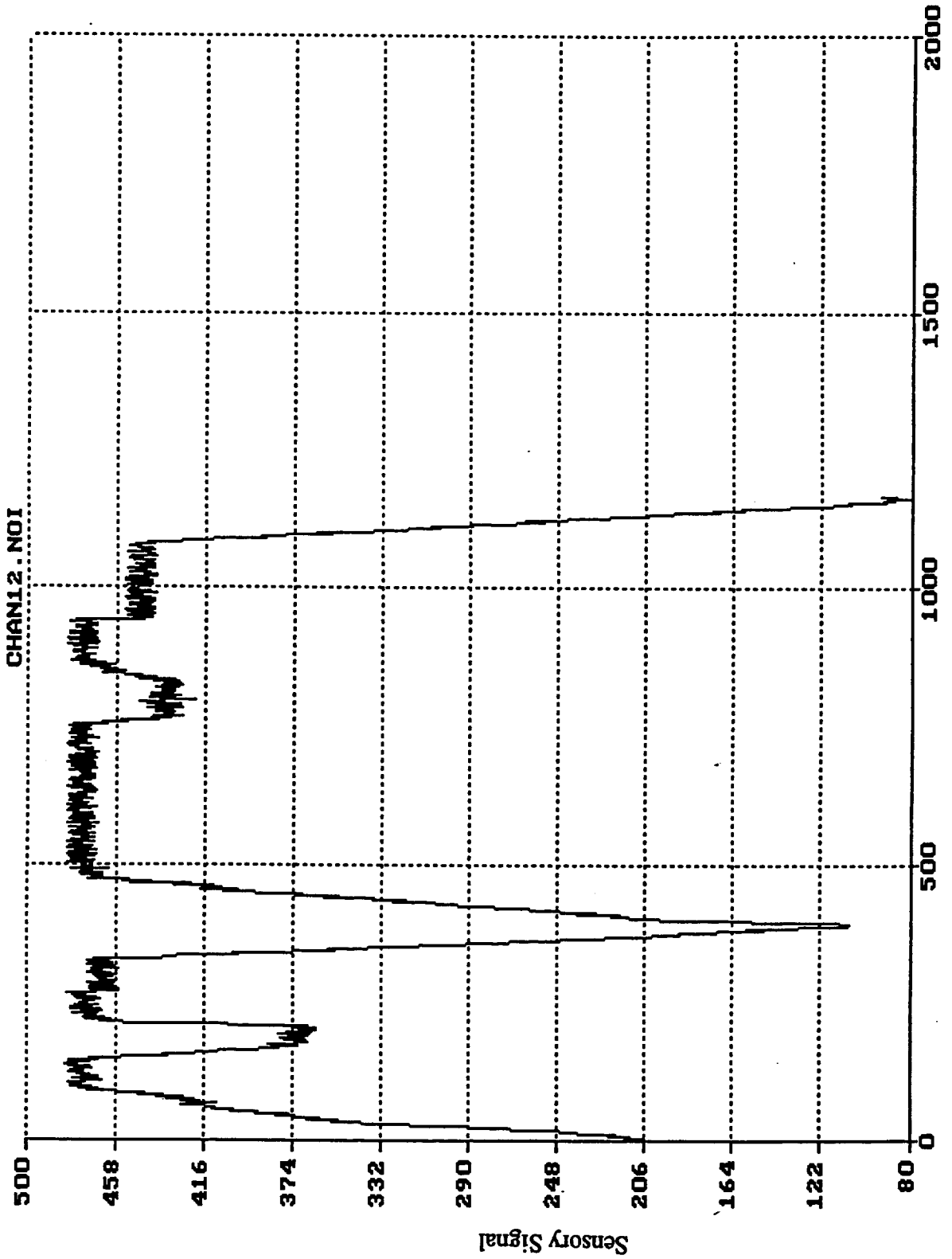

Fig.49: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for signal #12.

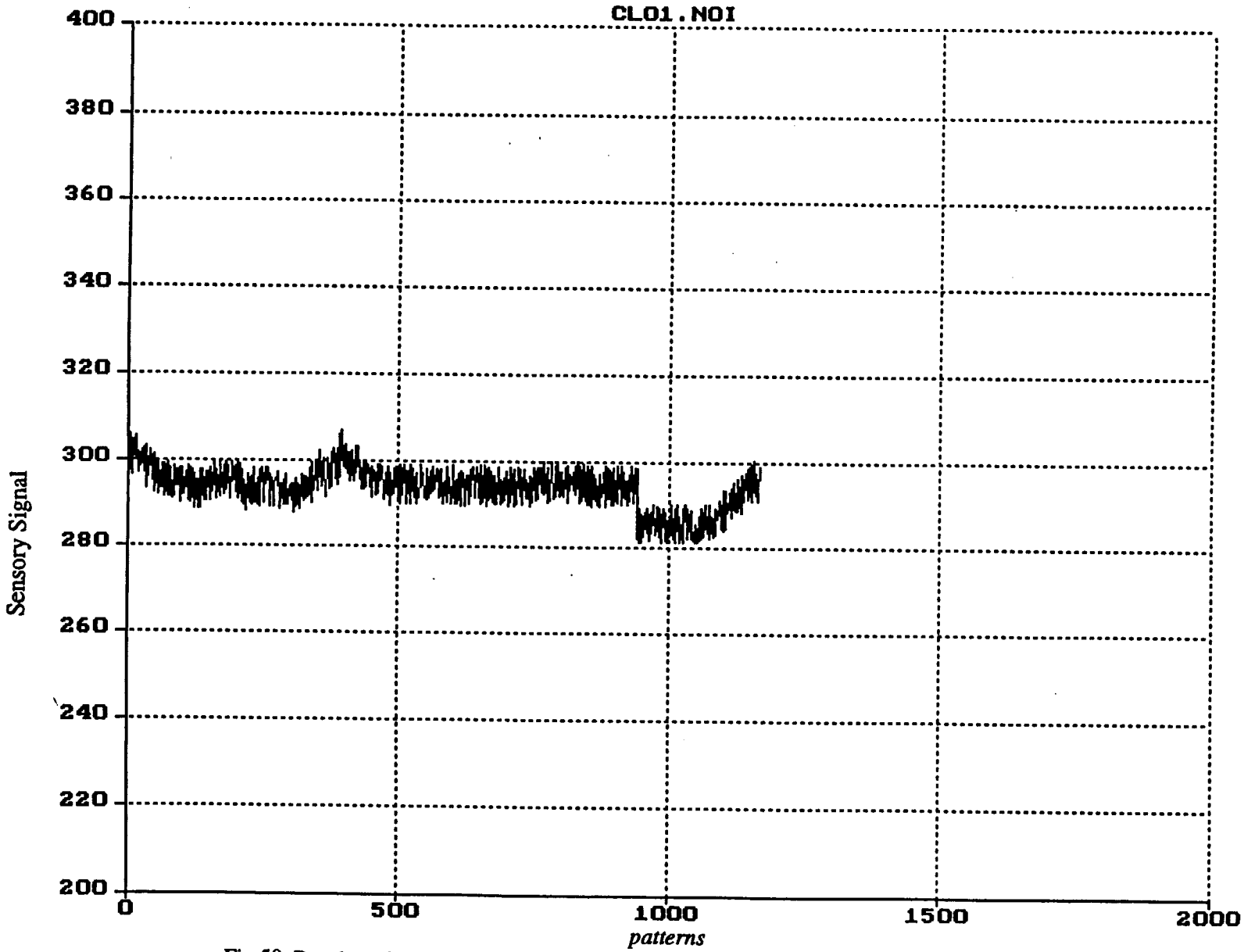

Fig.50 Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for loop-1 cold-leg temperature.

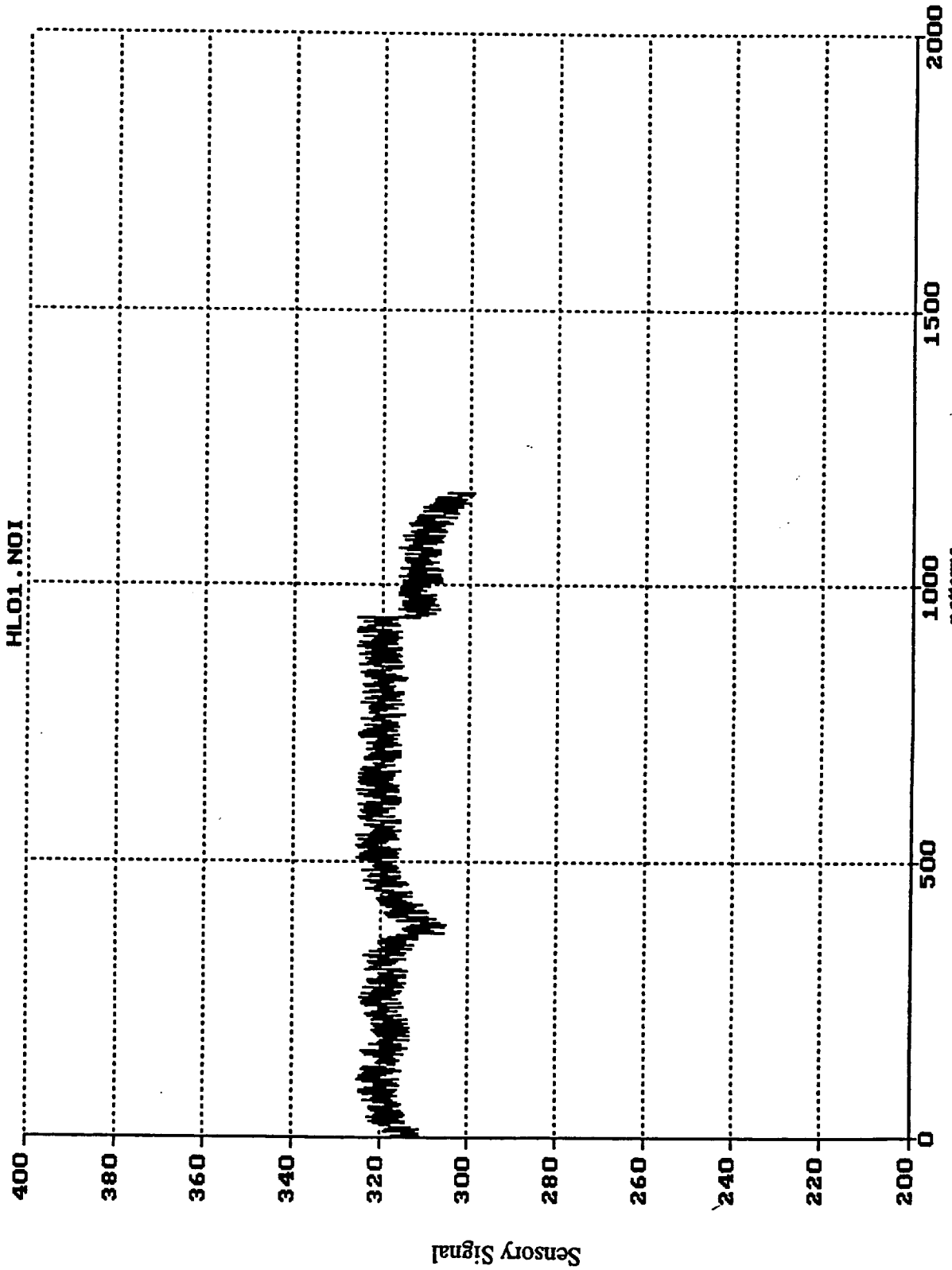

Fig.51: Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for loop-1 hot-leg temperature.

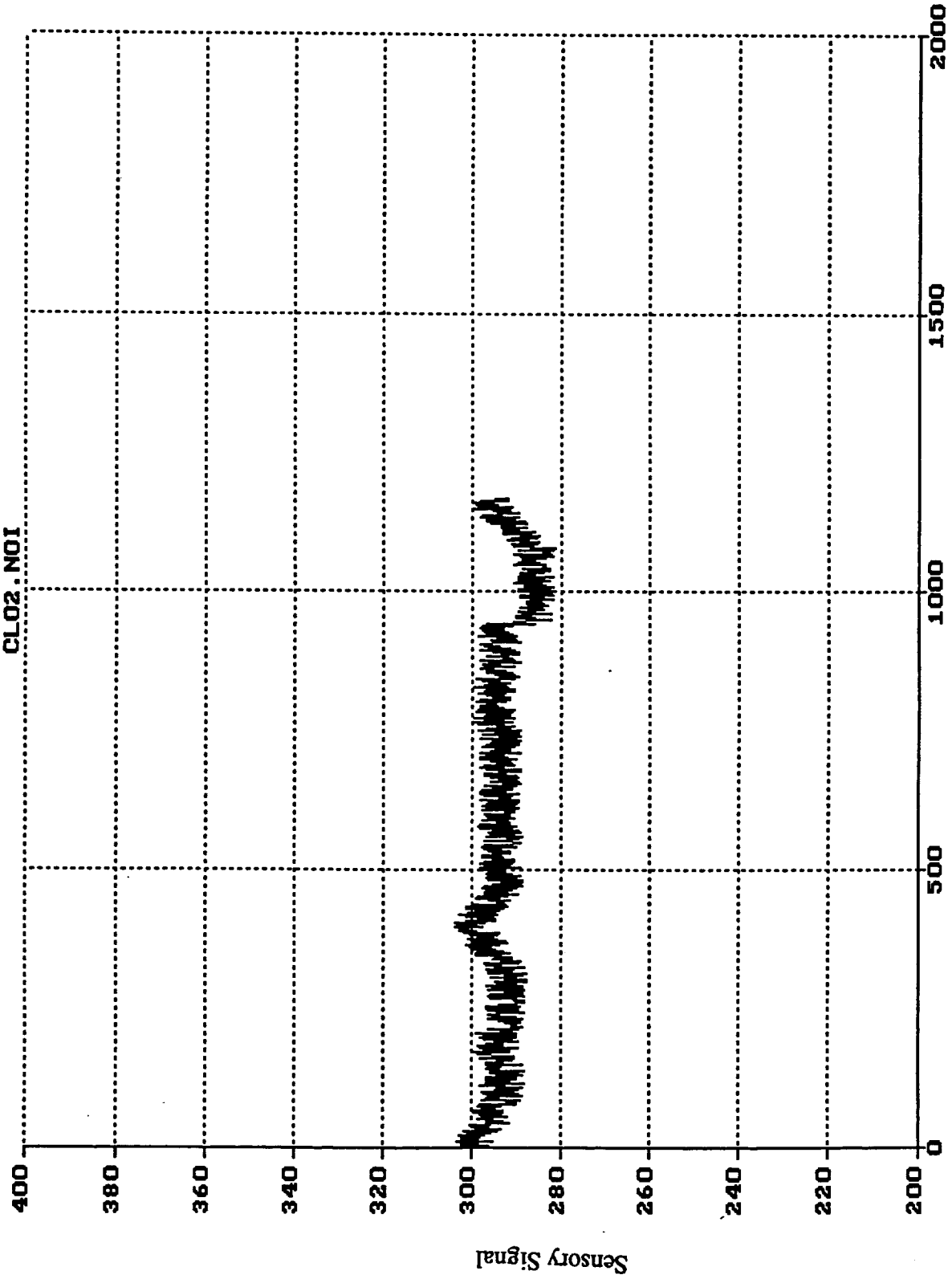

Fig.52 : Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for loop-2 cold-leg temperature.

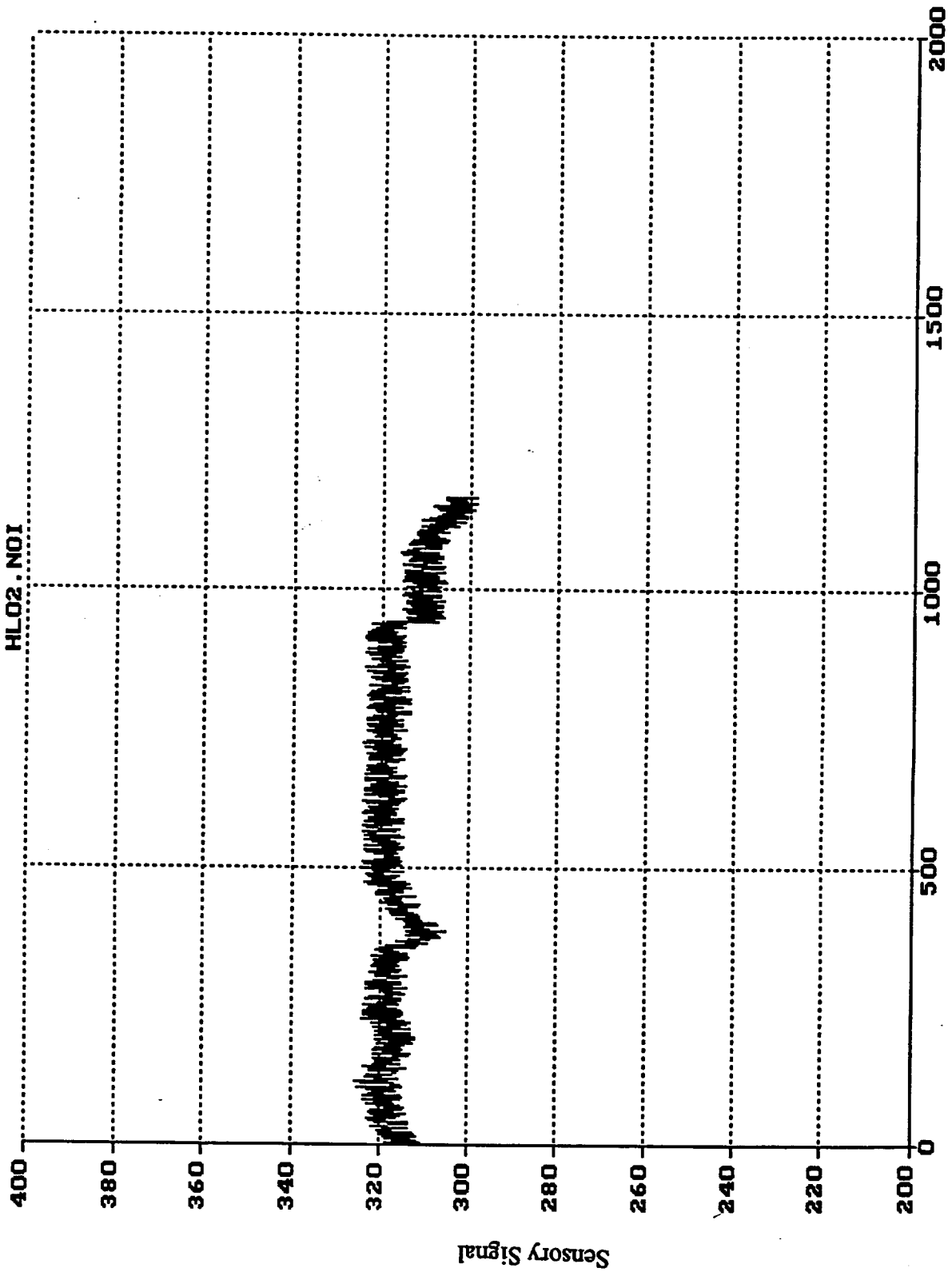

Fig.53 : Benchmark data with additive noise. The standard deviation of the noise is 3% of the maximum value of the signal among the total patterns and it is kept constant for that signal throughout the patterns. The representation is for loop-2 hot-leg temperature.

TABLE I : SENSITIVITY AND GRADIENT ANALYSIS OF THE TRAINED AUTO-ASSOCIATIVE NEURAL NETWORK USING THE BENCHMARK DATA

```
Benchmark data (B18B1.FRM)
Analysis results file(SAGRAD1.OUT)


L,(NH(I),I=0,IH),N>            12            8            12
 INP.CHANNELS :
  1   7   8  14  15  16  17  18  19  24  25  26  27  29

 OUT.CHANNELS :
  1   7   8  14  15  16  17  18  19  24  25  26  27  29

SENSITIVITY AND GRADIENT ANALYSIS IS GIVEN BELOW:

NORMALIZED GRADIENTS (I):

I=  1  II=  1 GN(I,II)=  0.674168
I=  2  II=  1 GN(I,II)=  0.158834
I=  3  II=  1 GN(I,II)=  0.490348
I=  4  II=  1 GN(I,II)=  1.000000
I=  5  II=  1 GN(I,II)=  0.484737
I=  6  II=  1 GN(I,II)=  0.413972
I=  7  II=  1 GN(I,II)=  0.329863
I=  8  II=  1 GN(I,II)=  0.220791
I=  9  II=  1 GN(I,II)=  0.136832
I= 10  II=  1 GN(I,II)=  0.682162
I= 11  II=  1 GN(I,II)=  0.863723
I= 12  II=  1 GN(I,II)=  0.781949
   1          I=  4  II=  1 GN(I,II)=  1.000000
   2          I= 11  II=  1 GN(I,II)=  0.863723
   3          I= 12  II=  1 GN(I,II)=  0.781949
   4          I= 10  II=  1 GN(I,II)=  0.682162
   5          I=  1  II=  1 GN(I,II)=  0.674168
   6          I=  3  II=  1 GN(I,II)=  0.490348
   7          I=  5  II=  1 GN(I,II)=  0.484737
   8          I=  6  II=  1 GN(I,II)=  0.413972
   9          I=  7  II=  1 GN(I,II)=  0.329863
  10          I=  8  II=  1 GN(I,II)=  0.220791
  11          I=  2  II=  1 GN(I,II)=  0.158834
  12          I=  9  II=  1 GN(I,II)=  0.136832


I=  1  II=  2 GN(I,II)=  0.765686
I=  2  II=  2 GN(I,II)=  0.216742
I=  3  II=  2 GN(I,II)=  0.386254
I=  4  II=  2 GN(I,II)=  1.000000
I=  5  II=  2 GN(I,II)=  0.569323
I=  6  II=  2 GN(I,II)=  0.598399
I=  7  II=  2 GN(I,II)=  0.466804
I=  8  II=  2 GN(I,II)=  0.223089
I=  9  II=  2 GN(I,II)=  0.148417
I= 10  II=  2 GN(I,II)=  0.591645
I= 11  II=  2 GN(I,II)=  0.665950
I= 12  II=  2 GN(I,II)=  0.554711
   1          I=  4  II=  2 GN(I,II)=  1.000000
   2          I=  1  II=  2 GN(I,II)=  0.765686
   3          I= 11  II=  2 GN(I,II)=  0.665950
   4          I=  6  II=  2 GN(I,II)=  0.598399
   5          I= 10  II=  2 GN(I,II)=  0.591645
   6          I=  5  II=  2 GN(I,II)=  0.569323
   7          I= 12  II=  2 GN(I,II)=  0.554711
   8          I=  7  II=  2 GN(I,II)=  0.466804
```

```
   9          I=  3  II=  2 GN(I,II)=  0.386254
  10          I=  8  II=  2 GN(I,II)=  0.223089
  11          I=  2  II=  2 GN(I,II)=  0.216742
  12          I=  9  II=  2 GN(I,II)=  0.148417


I=  1  II=  3 GN(I,II)=  0.693057
I=  2  II=  3 GN(I,II)=  0.225143
I=  3  II=  3 GN(I,II)=  0.414381
I=  4  II=  3 GN(I,II)=  1.000000
I=  5  II=  3 GN(I,II)=  0.534945
I=  6  II=  3 GN(I,II)=  0.572183
I=  7  II=  3 GN(I,II)=  0.408456
I=  8  II=  3 GN(I,II)=  0.232871
I=  9  II=  3 GN(I,II)=  0.154719
I= 10  II=  3 GN(I,II)=  0.584133
I= 11  II=  3 GN(I,II)=  0.720934
I= 12  II=  3 GN(I,II)=  0.724386
   1          I=  4  II=  3 GN(I,II)=  1.000000
   2          I= 12  II=  3 GN(I,II)=  0.724386
   3          I= 11  II=  3 GN(I,II)=  0.720934
   4          I=  1  II=  3 GN(I,II)=  0.693057
   5          I= 10  II=  3 GN(I,II)=  0.584133
   6          I=  6  II=  3 GN(I,II)=  0.572183
   7          I=  5  II=  3 GN(I,II)=  0.534945
   8          I=  3  II=  3 GN(I,II)=  0.414381
   9          I=  7  II=  3 GN(I,II)=  0.408456
  10          I=  8  II=  3 GN(I,II)=  0.232871
  11          I=  2  II=  3 GN(I,II)=  0.225143
  12          I=  9  II=  3 GN(I,II)=  0.154719


I=  1  II=  4 GN(I,II)=  0.688285
I=  2  II=  4 GN(I,II)=  0.127667
I=  3  II=  4 GN(I,II)=  0.413368
I=  4  II=  4 GN(I,II)=  0.877243
I=  5  II=  4 GN(I,II)=  0.406592
I=  6  II=  4 GN(I,II)=  0.520944
I=  7  II=  4 GN(I,II)=  0.384686
I=  8  II=  4 GN(I,II)=  0.197231
I=  9  II=  4 GN(I,II)=  0.146365
I= 10  II=  4 GN(I,II)=  0.784480
I= 11  II=  4 GN(I,II)=  1.000000
I= 12  II=  4 GN(I,II)=  0.740037
   1          I= 11  II=  4 GN(I,II)=  1.000000
   2          I=  4  II=  4 GN(I,II)=  0.877243
   3          I= 10  II=  4 GN(I,II)=  0.784480
   4          I= 12  II=  4 GN(I,II)=  0.740037
   5          I=  1  II=  4 GN(I,II)=  0.688285
   6          I=  6  II=  4 GN(I,II)=  0.520944
   7          I=  3  II=  4 GN(I,II)=  0.413368
   8          I=  5  II=  4 GN(I,II)=  0.406592
   9          I=  7  II=  4 GN(I,II)=  0.384686
  10          I=  8  II=  4 GN(I,II)=  0.197231
  11          I=  9  II=  4 GN(I,II)=  0.146365
  12          I=  2  II=  4 GN(I,II)=  0.127667


I=  1  II=  5 GN(I,II)=  0.809163
I=  2  II=  5 GN(I,II)=  0.162190
I=  3  II=  5 GN(I,II)=  0.281448
```

```
I=  4  II=  5 GN(I,II)=  1.000000
I=  5  II=  5 GN(I,II)=  0.535838
I=  6  II=  5 GN(I,II)=  0.612962
I=  7  II=  5 GN(I,II)=  0.466676
I=  8  II=  5 GN(I,II)=  0.180707
I=  9  II=  5 GN(I,II)=  0.162352
I= 10  II=  5 GN(I,II)=  0.661936
I= 11  II=  5 GN(I,II)=  0.805150
I= 12  II=  5 GN(I,II)=  0.607527
   1          I=  4  II=  5 GN(I,II)=  1.000000
   2          I=  1  II=  5 GN(I,II)=  0.809163
   3          I= 11  II=  5 GN(I,II)=  0.805150
   4          I= 10  II=  5 GN(I,II)=  0.661936
   5          I=  6  II=  5 GN(I,II)=  0.612962
   6          I= 12  II=  5 GN(I,II)=  0.607527
   7          I=  5  II=  5 GN(I,II)=  0.535838
   8          I=  7  II=  5 GN(I,II)=  0.466676
   9          I=  3  II=  5 GN(I,II)=  0.281448
  10          I=  8  II=  5 GN(I,II)=  0.180707
  11          I=  9  II=  5 GN(I,II)=  0.162352
  12          I=  2  II=  5 GN(I,II)=  0.162190


I=  1  II=  6 GN(I,II)=  0.716446
I=  2  II=  6 GN(I,II)=  0.169959
I=  3  II=  6 GN(I,II)=  0.404861
I=  4  II=  6 GN(I,II)=  1.000000
I=  5  II=  6 GN(I,II)=  0.465642
I=  6  II=  6 GN(I,II)=  0.513893
I=  7  II=  6 GN(I,II)=  0.368502
I=  8  II=  6 GN(I,II)=  0.220955
I=  9  II=  6 GN(I,II)=  0.145357
I= 10  II=  6 GN(I,II)=  0.690590
I= 11  II=  6 GN(I,II)=  0.868702
I= 12  II=  6 GN(I,II)=  0.769814
   1          I=  4  II=  6 GN(I,II)=  1.000000
   2          I= 11  II=  6 GN(I,II)=  0.868702
   3          I= 12  II=  6 GN(I,II)=  0.769814
   4          I=  1  II=  6 GN(I,II)=  0.716446
   5          I= 10  II=  6 GN(I,II)=  0.690590
   6          I=  6  II=  6 GN(I,II)=  0.513893
   7          I=  5  II=  6 GN(I,II)=  0.465642
   8          I=  3  II=  6 GN(I,II)=  0.404861
   9          I=  7  II=  6 GN(I,II)=  0.368502
  10          I=  8  II=  6 GN(I,II)=  0.220955
  11          I=  2  II=  6 GN(I,II)=  0.169959
  12          I=  9  II=  6 GN(I,II)=  0.145357


I=  1  II=  7 GN(I,II)=  0.734108
I=  2  II=  7 GN(I,II)=  0.174696
I=  3  II=  7 GN(I,II)=  0.417865
I=  4  II=  7 GN(I,II)=  1.000000
I=  5  II=  7 GN(I,II)=  0.514429
I=  6  II=  7 GN(I,II)=  0.511206
I=  7  II=  7 GN(I,II)=  0.385923
I=  8  II=  7 GN(I,II)=  0.227886
I=  9  II=  7 GN(I,II)=  0.153302
I= 10  II=  7 GN(I,II)=  0.699806
I= 11  II=  7 GN(I,II)=  0.869809
I= 12  II=  7 GN(I,II)=  0.741906
   1          I=  4  II=  7 GN(I,II)=  1.000000
```

```
   2          I= 11  II=  7 GN(I,II)=  0.869809
   3          I= 12  II=  7 GN(I,II)=  0.741906
   4          I=  1  II=  7 GN(I,II)=  0.734108
   5          I= 10  II=  7 GN(I,II)=  0.699806
   6          I=  5  II=  7 GN(I,II)=  0.514429
   7          I=  6  II=  7 GN(I,II)=  0.511206
   8          I=  3  II=  7 GN(I,II)=  0.417865
   9          I=  7  II=  7 GN(I,II)=  0.385923
  10          I=  8  II=  7 GN(I,II)=  0.227886
  11          I=  2  II=  7 GN(I,II)=  0.174696
  12          I=  9  II=  7 GN(I,II)=  0.153302


 I=  1  II=  8 GN(I,II)=  0.765821
 I=  2  II=  8 GN(I,II)=  0.206510
 I=  3  II=  8 GN(I,II)=  0.336093
 I=  4  II=  8 GN(I,II)=  1.000000
 I=  5  II=  8 GN(I,II)=  0.512773
 I=  6  II=  8 GN(I,II)=  0.600043
 I=  7  II=  8 GN(I,II)=  0.448867
 I=  8  II=  8 GN(I,II)=  0.216773
 I=  9  II=  8 GN(I,II)=  0.138847
 I= 10  II=  8 GN(I,II)=  0.576596
 I= 11  II=  8 GN(I,II)=  0.641232
 I= 12  II=  8 GN(I,II)=  0.547676
   1          I=  4  II=  8 GN(I,II)=  1.000000
   2          I=  1  II=  8 GN(I,II)=  0.765821
   3          I= 11  II=  8 GN(I,II)=  0.641232
   4          I=  6  II=  8 GN(I,II)=  0.600043
   5          I= 10  II=  8 GN(I,II)=  0.576596
   6          I= 12  II=  8 GN(I,II)=  0.547676
   7          I=  5  II=  8 GN(I,II)=  0.512773
   8          I=  7  II=  8 GN(I,II)=  0.448867
   9          I=  3  II=  8 GN(I,II)=  0.336093
  10          I=  8  II=  8 GN(I,II)=  0.216773
  11          I=  2  II=  8 GN(I,II)=  0.206510
  12          I=  9  II=  8 GN(I,II)=  0.138847


 I=  1  II=  9 GN(I,II)=  0.753901
 I=  2  II=  9 GN(I,II)=  0.206203
 I=  3  II=  9 GN(I,II)=  0.324172
 I=  4  II=  9 GN(I,II)=  1.000000
 I=  5  II=  9 GN(I,II)=  0.506114
 I=  6  II=  9 GN(I,II)=  0.592465
 I=  7  II=  9 GN(I,II)=  0.436632
 I=  8  II=  9 GN(I,II)=  0.211385
 I=  9  II=  9 GN(I,II)=  0.136552
 I= 10  II=  9 GN(I,II)=  0.556240
 I= 11  II=  9 GN(I,II)=  0.625210
 I= 12  II=  9 GN(I,II)=  0.557346
   1          I=  4  II=  9 GN(I,II)=  1.000000
   2          I=  1  II=  9 GN(I,II)=  0.753901
   3          I= 11  II=  9 GN(I,II)=  0.625210
   4          I=  6  II=  9 GN(I,II)=  0.592465
   5          I= 12  II=  9 GN(I,II)=  0.557346
   6          I= 10  II=  9 GN(I,II)=  0.556240
   7          I=  5  II=  9 GN(I,II)=  0.506114
   8          I=  7  II=  9 GN(I,II)=  0.436632
   9          I=  3  II=  9 GN(I,II)=  0.324172
  10          I=  8  II=  9 GN(I,II)=  0.211385
  11          I=  2  II=  9 GN(I,II)=  0.206203
```

```
  12          I=  9  II=  9 GN(I,II)=  0.136552


I=  1  II= 10 GN(I,II)=  0.652083
I=  2  II= 10 GN(I,II)=  0.181219
I=  3  II= 10 GN(I,II)=  0.464236
I=  4  II= 10 GN(I,II)=  0.933169
I=  5  II= 10 GN(I,II)=  0.507815
I=  6  II= 10 GN(I,II)=  0.480354
I=  7  II= 10 GN(I,II)=  0.331138
I=  8  II= 10 GN(I,II)=  0.251192
I=  9  II= 10 GN(I,II)=  0.169727
I= 10  II= 10 GN(I,II)=  0.747565
I= 11  II= 10 GN(I,II)=  1.000000
I= 12  II= 10 GN(I,II)=  0.906897
   1          I= 11  II= 10 GN(I,II)=  1.000000
   2          I=  4  II= 10 GN(I,II)=  0.933169
   3          I= 12  II= 10 GN(I,II)=  0.906897
   4          I= 10  II= 10 GN(I,II)=  0.747565
   5          I=  1  II= 10 GN(I,II)=  0.652083
   6          I=  5  II= 10 GN(I,II)=  0.507815
   7          I=  6  II= 10 GN(I,II)=  0.480354
   8          I=  3  II= 10 GN(I,II)=  0.464236
   9          I=  7  II= 10 GN(I,II)=  0.331138
  10          I=  8  II= 10 GN(I,II)=  0.251192
  11          I=  2  II= 10 GN(I,II)=  0.181219
  12          I=  9  II= 10 GN(I,II)=  0.169727


I=  1  II= 11 GN(I,II)=  0.630699
I=  2  II= 11 GN(I,II)=  0.169912
I=  3  II= 11 GN(I,II)=  0.449660
I=  4  II= 11 GN(I,II)=  0.868861
I=  5  II= 11 GN(I,II)=  0.465252
I=  6  II= 11 GN(I,II)=  0.489645
I=  7  II= 11 GN(I,II)=  0.335210
I=  8  II= 11 GN(I,II)=  0.241737
I=  9  II= 11 GN(I,II)=  0.162877
I= 10  II= 11 GN(I,II)=  0.754446
I= 11  II= 11 GN(I,II)=  1.000000
I= 12  II= 11 GN(I,II)=  0.859477
   1          I= 11  II= 11 GN(I,II)=  1.000000
   2          I=  4  II= 11 GN(I,II)=  0.868861
   3          I= 12  II= 11 GN(I,II)=  0.859477
   4          I= 10  II= 11 GN(I,II)=  0.754446
   5          I=  1  II= 11 GN(I,II)=  0.630699
   6          I=  6  II= 11 GN(I,II)=  0.489645
   7          I=  5  II= 11 GN(I,II)=  0.465252
   8          I=  3  II= 11 GN(I,II)=  0.449660
   9          I=  7  II= 11 GN(I,II)=  0.335210
  10          I=  8  II= 11 GN(I,II)=  0.241737
  11          I=  2  II= 11 GN(I,II)=  0.169912
  12          I=  9  II= 11 GN(I,II)=  0.162877


I=  1  II= 12 GN(I,II)=  0.600956
I=  2  II= 12 GN(I,II)=  0.179769
I=  3  II= 12 GN(I,II)=  0.495811
I=  4  II= 12 GN(I,II)=  0.928283
I=  5  II= 12 GN(I,II)=  0.425348
I=  6  II= 12 GN(I,II)=  0.506188
```

```
I=  7  II= 12 GN(I,II)=  0.327291
I=  8  II= 12 GN(I,II)=  0.232816
I=  9  II= 12 GN(I,II)=  0.158494
I= 10  II= 12 GN(I,II)=  0.730058
I= 11  II= 12 GN(I,II)=  1.000000
I= 12  II= 12 GN(I,II)=  0.962578
   1          I= 11  II= 12 GN(I,II)=  1.000000
   2          I= 12  II= 12 GN(I,II)=  0.962578
   3          I=  4  II= 12 GN(I,II)=  0.928283
   4          I= 10  II= 12 GN(I,II)=  0.730058
   5          I=  1  II= 12 GN(I,II)=  0.600956
   6          I=  6  II= 12 GN(I,II)=  0.506188
   7          I=  3  II= 12 GN(I,II)=  0.495811
   8          I=  5  II= 12 GN(I,II)=  0.425348
   9          I=  7  II= 12 GN(I,II)=  0.327291
  10          I=  8  II= 12 GN(I,II)=  0.232816
  11          I=  2  II= 12 GN(I,II)=  0.179769
  12          I=  9  II= 12 GN(I,II)=  0.158494


 SENS END

 GRAD START:
   0       0.7340E-03 -0.1233E-03  0.1261E-03  0.2153E-03 -0.2386E-03
   5      -0.2073E-03  0.7142E-03  0.6177E-04  0.3092E-03 -0.1011E-02
  10       0.2424E-03  0.2382E-03  0.1036E-07  0.4470E-06  0.6528E-05
  15      -0.3128E-06 -0.9481E-06  0.4638E-06 -0.5043E-05 -0.1709E-05
  20      -0.1540E-05  0.5261E-05 -0.2896E-05  0.7952E-06 -0.6737E-03
  25       0.3769E-03 -0.1953E-03 -0.1015E-03  0.7940E-04 -0.2602E-04
  30      -0.3974E-03 -0.3602E-03 -0.8111E-05  0.8543E-03  0.3404E-04
  35      -0.1468E-03 -0.6260E-04 -0.2492E-04  0.1290E-03 -0.4740E-03
  40       0.1473E-03  0.1801E-03 -0.7399E-05  0.1043E-04  0.7183E-04
  45       0.2460E-03 -0.1561E-03 -0.1996E-03 -0.5650E-03 -0.4701E-03
  50      -0.3241E-03 -0.1805E-03  0.6701E-04  0.1584E-03  0.1072E-04
  55      -0.4846E-04  0.3099E-04  0.4635E-03 -0.1969E-03 -0.1990E-03
  60      -0.2173E-03 -0.1437E-03 -0.1038E-03 -0.1951E-03  0.1237E-03
  65       0.2853E-03 -0.1976E-03 -0.6916E-04  0.2961E-03  0.1023E-03
  70      -0.9028E-04 -0.6853E-04 -0.3157E-03 -0.3848E-03  0.1491E-03
  75      -0.2498E-03  0.1200E-03  0.5846E-04 -0.3265E-03  0.8080E-04
  80      -0.1723E-03  0.5886E-03 -0.2007E-03 -0.2081E-03 -0.1793E-03
  85       0.3842E-03 -0.2743E-04 -0.2736E-03  0.1130E-03  0.3726E-03
  90      -0.8722E-04 -0.2982E-03 -0.4486E-03  0.3514E-03  0.6241E-04
  95      -0.2269E-03 -0.5551E-03 -0.5998E-04 -0.9980E-04 -0.9773E-04
 100       0.1155E-03 -0.1761E-03 -0.1168E-03 -0.1277E-03 -0.2799E-03
 105       0.5391E-03 -0.3986E-04 -0.5441E-05  0.2484E-04 -0.3233E-04
 110       0.7745E-03  0.6298E-04 -0.2834E-03  0.4235E-03 -0.6396E-03
 115      -0.3153E-03  0.9540E-05  0.6628E-05  0.7706E-05 -0.7621E-06
 120      -0.1886E-05  0.4091E-05 -0.1568E-04 -0.1273E-05  0.1220E-06
 125      -0.5138E-05 -0.6325E-06 -0.2854E-05 -0.4316E-03  0.2536E-03
 130       0.2160E-03 -0.3573E-03 -0.1842E-03 -0.2781E-03 -0.3092E-03
 135       0.2641E-03  0.2828E-03 -0.3496E-03 -0.3282E-03 -0.3168E-03
 140      -0.6320E-03  0.5674E-03  0.5567E-03 -0.5717E-03 -0.2929E-03
 145      -0.3592E-03 -0.6183E-03  0.5156E-03  0.4868E-03 -0.7959E-03
 150      -0.7471E-03 -0.8100E-03 -0.1070E-03  0.4550E-04  0.5791E-04
 155      -0.9883E-04 -0.6788E-05 -0.1399E-03  0.4695E-04 -0.4677E-04
 160      -0.8984E-04 -0.1192E-03 -0.3173E-04  0.1075E-04  0.1433E-03
 165      -0.9419E-04  0.2385E-04 -0.1127E-03 -0.4161E-04 -0.1426E-03
 170       0.2173E-03  0.1134E-04  0.4064E-04  0.6151E-04 -0.7163E-04
 175      -0.7442E-05 -0.1652E-03  0.4014E-03  0.4045E-03 -0.2522E-03
 180      -0.3378E-03 -0.1643E-03 -0.2330E-03  0.3305E-03  0.3653E-03
 185      -0.4613E-04 -0.9499E-04 -0.3830E-03  0.2962E-03 -0.2017E-03
 190      -0.1714E-03  0.4563E-04  0.1493E-04  0.1072E-03  0.2149E-03
 195      -0.1892E-03 -0.1781E-03  0.2200E-03  0.7894E-04  0.2201E-03
```

```
200        0.1701E-03 -0.2966E-03 -0.2779E-03  0.4045E-03  0.3031E-03
205        0.2701E-03  0.1653E-03 -0.2504E-03 -0.2266E-03  0.1694E-03
210        0.2340E-03  0.2120E-03  0.0000E+00  0.0000E+00  0.0000E+00


output layer biases:

K,GG,=            1    7.340052E-04
K,GG,=            2   -1.233192E-04
K,GG,=            3    1.261068E-04
K,GG,=            4    2.153236E-04
K,GG,=            5   -2.386288E-04
K,GG,=            6   -2.073063E-04
K,GG,=            7    7.142138E-04
K,GG,=            8    6.176558E-05
K,GG,=            9    3.091907E-04
K,GG,=           10   -1.010862E-03
K,GG,=           11    2.424035E-04
K,GG,=           12    2.381626E-04
 hidden-to-output layer weights:

K,GG=            13    1.035848E-08
K,GG=            14    4.470400E-07
K,GG=            15    6.528255E-06
K,GG=            16   -3.127620E-07
K,GG=            17   -9.480654E-07
K,GG=            18    4.638500E-07
K,GG=            19   -5.043093E-06
K,GG=            20   -1.709260E-06
K,GG=            21   -1.539706E-06
K,GG=            22    5.261207E-06
K,GG=            23   -2.895923E-06
K,GG=            24    7.951830E-07
K,GG=            25   -6.737274E-04
K,GG=            26    3.769007E-04
K,GG=            27   -1.952790E-04
K,GG=            28   -1.015100E-04
K,GG=            29    7.939786E-05
K,GG=            30   -2.601687E-05
K,GG=            31   -3.974412E-04
K,GG=            32   -3.602385E-04
K,GG=            33   -8.111444E-06
K,GG=            34    8.543016E-04
K,GG=            35    3.403830E-05
K,GG=            36   -1.467511E-04
K,GG=            37   -6.260371E-05
K,GG=            38   -2.492352E-05
K,GG=            39    1.290315E-04
K,GG=            40   -4.739946E-04
K,GG=            41    1.472927E-04
K,GG=            42    1.800790E-04
K,GG=            43   -7.398644E-06
K,GG=            44    1.042680E-05
K,GG=            45    7.182884E-05
K,GG=            46    2.460130E-04
K,GG=            47   -1.561105E-04
K,GG=            48   -1.996298E-04
K,GG=            49   -5.649998E-04
K,GG=            50   -4.700771E-04
K,GG=            51   -3.240532E-04
K,GG=            52   -1.804544E-04
K,GG=            53    6.701439E-05
K,GG=            54    1.583739E-04
```

```
K,GG=          55    1.071916E-05
K,GG=          56   -4.846122E-05
K,GG=          57    3.099180E-05
K,GG=          58    4.634827E-04
K,GG=          59   -1.968731E-04
K,GG=          60   -1.989590E-04
K,GG=          61   -2.173436E-04
K,GG=          62   -1.437469E-04
K,GG=          63   -1.037573E-04
K,GG=          64   -1.950556E-04
K,GG=          65    1.236867E-04
K,GG=          66    2.852633E-04
K,GG=          67   -1.975981E-04
K,GG=          68   -6.915593E-05
K,GG=          69    2.961299E-04
K,GG=          70    1.022922E-04
K,GG=          71   -9.028285E-05
K,GG=          72   -6.853251E-05
K,GG=          73   -3.156559E-04
K,GG=          74   -3.848452E-04
K,GG=          75    1.491408E-04
K,GG=          76   -2.498289E-04
K,GG=          77    1.199933E-04
K,GG=          78    5.846245E-05
K,GG=          79   -3.265305E-04
K,GG=          80    8.080386E-05
K,GG=          81   -1.722585E-04
K,GG=          82    5.885922E-04
K,GG=          83   -2.006928E-04
K,GG=          84   -2.080719E-04
K,GG=          85   -1.792769E-04
K,GG=          86    3.841805E-04
K,GG=          87   -2.742545E-05
K,GG=          88   -2.735973E-04
K,GG=          89    1.129895E-04
K,GG=          90    3.725638E-04
K,GG=          91   -8.721712E-05
K,GG=          92   -2.981594E-04
K,GG=          93   -4.486237E-04
K,GG=          94    3.514400E-04
K,GG=          95    6.241397E-05
K,GG=          96   -2.268905E-04
K,GG=          97   -5.551148E-04
K,GG=          98   -5.997965E-05
K,GG=          99   -9.979542E-05
K,GG=         100   -9.772792E-05
K,GG=         101    1.155172E-04
K,GG=         102   -1.760644E-04
K,GG=         103   -1.168164E-04
K,GG=         104   -1.277295E-04
K,GG=         105   -2.799164E-04
K,GG=         106    5.391492E-04
K,GG=         107   -3.985829E-05
K,GG=         108   -5.441226E-06

EXTRA HIDDEN LAYER BIASES (1ST FROM END:
K,GG=         109    2.483544E-05
K,GG=         110   -3.233136E-05
K,GG=         111    7.745087E-04
K,GG=         112    6.298485E-05
K,GG=         113   -2.834333E-04
K,GG=         114    4.235221E-04
K,GG=         115   -6.396368E-04
K,GG=         116   -3.152992E-04
```

```
input-to-hidden layer weights:
 (( J=1,L), I=1,M) ; L,M=           12           8
K,GG=          117    9.539763E-06
K,GG=          118    6.627609E-06
K,GG=          119    7.705712E-06
K,GG=          120   -7.621233E-07
K,GG=          121   -1.885758E-06
K,GG=          122    4.091496E-06
K,GG=          123   -1.567569E-05
K,GG=          124   -1.272950E-06
K,GG=          125    1.219696E-07
K,GG=          126   -5.138312E-06
K,GG=          127   -6.324756E-07
K,GG=          128   -2.853571E-06
K,GG=          129   -4.315570E-04
K,GG=          130    2.536459E-04
K,GG=          131    2.160064E-04
K,GG=          132   -3.572800E-04
K,GG=          133   -1.841835E-04
K,GG=          134   -2.780837E-04
K,GG=          135   -3.092292E-04
K,GG=          136    2.641023E-04
K,GG=          137    2.827781E-04
K,GG=          138   -3.495761E-04
K,GG=          139   -3.282251E-04
K,GG=          140   -3.167633E-04
K,GG=          141   -6.319982E-04
K,GG=          142    5.674210E-04
K,GG=          143    5.566679E-04
K,GG=          144   -5.717376E-04
K,GG=          145   -2.928817E-04
K,GG=          146   -3.592413E-04
K,GG=          147   -6.182623E-04
K,GG=          148    5.156100E-04
K,GG=          149    4.867712E-04
K,GG=          150   -7.959353E-04
K,GG=          151   -7.471047E-04
K,GG=          152   -8.099733E-04
K,GG=          153   -1.069988E-04
K,GG=          154    4.550124E-05
K,GG=          155    5.791365E-05
K,GG=          156   -9.883454E-05
K,GG=          157   -6.787819E-06
K,GG=          158   -1.399114E-04
K,GG=          159    4.695045E-05
K,GG=          160   -4.676700E-05
K,GG=          161   -8.984478E-05
K,GG=          162   -1.191705E-04
K,GG=          163   -3.173420E-05
K,GG=          164    1.074753E-05
K,GG=          165    1.433013E-04
K,GG=          166   -9.418761E-05
K,GG=          167    2.385414E-05
K,GG=          168   -1.127278E-04
K,GG=          169   -4.160768E-05
K,GG=          170   -1.426428E-04
K,GG=          171    2.172890E-04
K,GG=          172    1.133806E-05
K,GG=          173    4.064025E-05
K,GG=          174    6.150801E-05
K,GG=          175   -7.162749E-05
K,GG=          176   -7.441570E-06
```

K,GG=	177	-1.651848E-04
K,GG=	178	4.014127E-04
K,GG=	179	4.044795E-04
K,GG=	180	-2.522087E-04
K,GG=	181	-3.377703E-04
K,GG=	182	-1.643384E-04
K,GG=	183	-2.330101E-04
K,GG=	184	3.305309E-04
K,GG=	185	3.652671E-04
K,GG=	186	-4.613128E-05
K,GG=	187	-9.498690E-05
K,GG=	188	-3.830182E-04
K,GG=	189	2.962159E-04
K,GG=	190	-2.017303E-04
K,GG=	191	-1.714081E-04
K,GG=	192	4.562850E-05
K,GG=	193	1.492751E-05
K,GG=	194	1.071706E-04
K,GG=	195	2.149171E-04
K,GG=	196	-1.891815E-04
K,GG=	197	-1.781057E-04
K,GG=	198	2.200444E-04
K,GG=	199	7.894166E-05
K,GG=	200	2.200658E-04
K,GG=	201	1.700727E-04
K,GG=	202	-2.966236E-04
K,GG=	203	-2.778864E-04
K,GG=	204	4.044805E-04
K,GG=	205	3.030678E-04
K,GG=	206	2.701117E-04
K,GG=	207	1.653074E-04
K,GG=	208	-2.504165E-04
K,GG=	209	-2.266463E-04
K,GG=	210	1.693592E-04
K,GG=	211	2.339673E-04
K,GG=	212	2.120160E-04

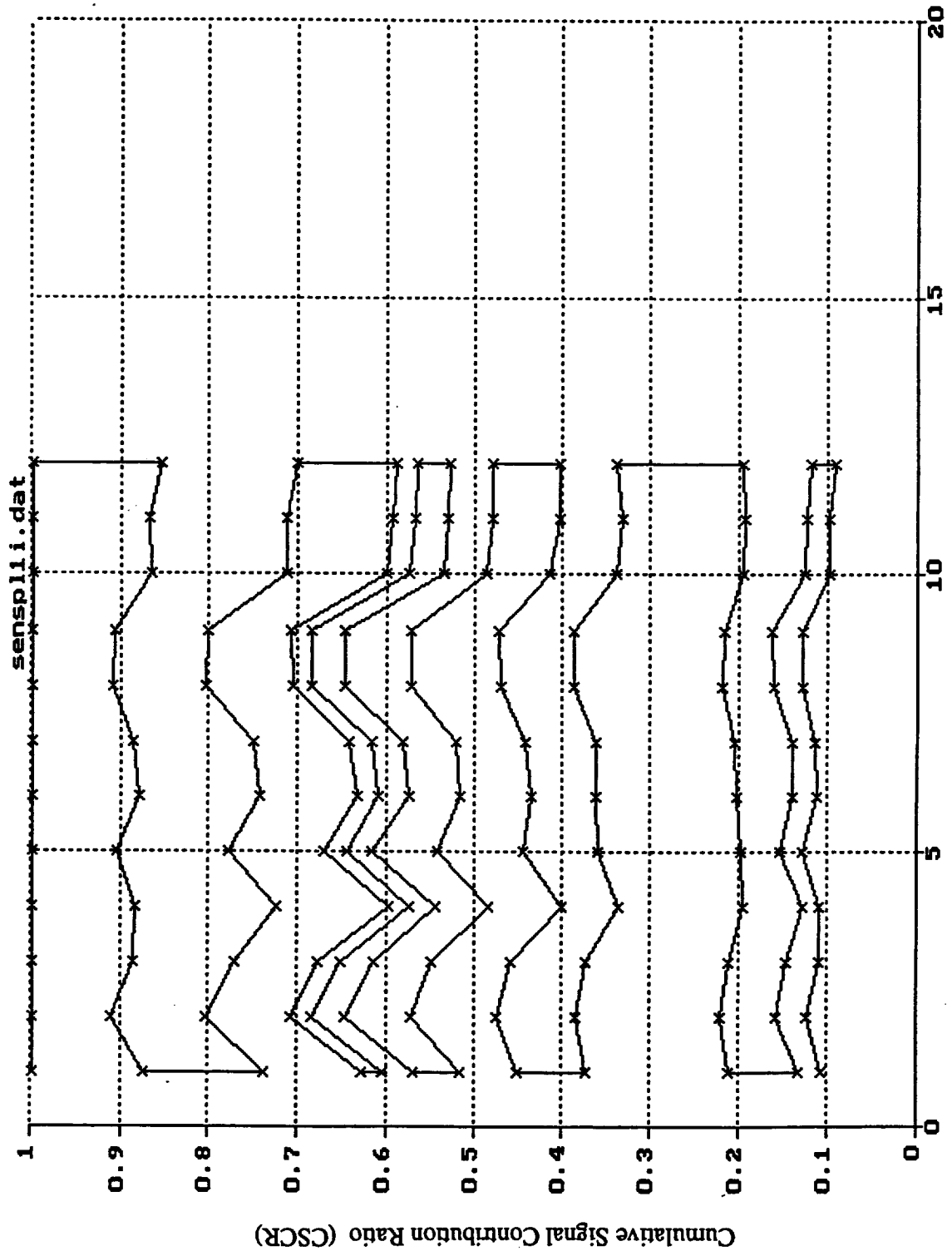

Fig.54 : Sensitivity analysis without additive noise. The horizontal-axis represents the output nodes. Each relative contribution to each output node is represented on the vertical-axis. The conributions are in cumulative form so that it amounts to unity for all signals. The network is autoassociative and the learning algorithm is variable-metric.

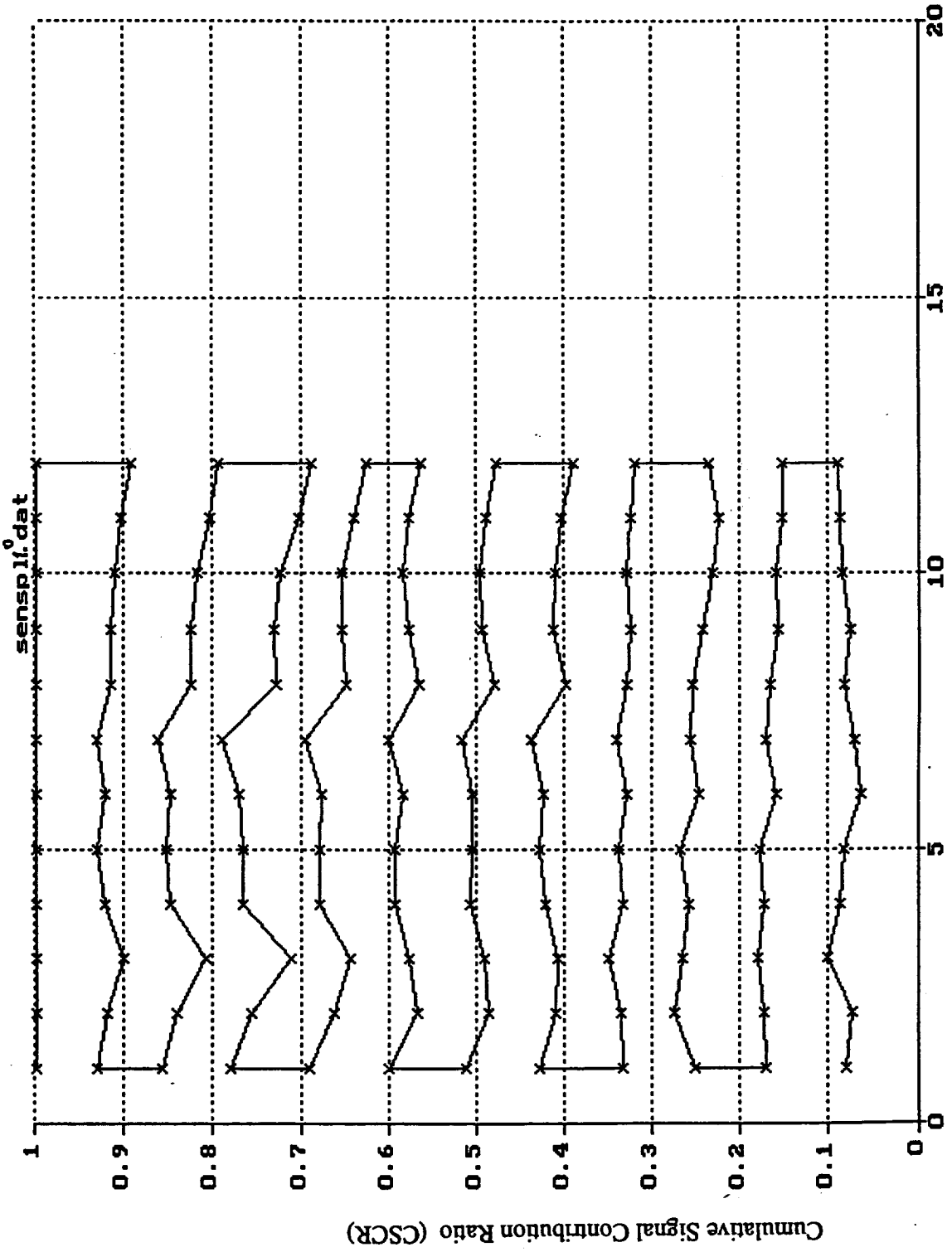

Fig.55 : Sensitivity analysis without additive noise. The horizontal-axis represents the input signals. Relative contribution of each input signal to all output nodes is represented on the vertical-axis. The contributions are in cumulative form so that it amounts to unity for all signals. The network is autoassociative and the learning algorithm is variable-metric.

TABLE II : SENSITIVITY AND GRADIENT ANALYSIS OF THE TRAINED AUTO-ASSOCIATIVE NEURAL NETWORK USING THE BENCHMARK DATA WITH ADDITIVE NOISE

```
Noisy benchmark data (BNCHN03.FRM)
Analysis results file(SAGRADN.OUT)


L,(NH(I),I=0,IH),N>            12            8            12
 INP.CHANNELS :
  1   7   8  14  15  16  17  18  19  24  25  26  27  29

 OUT.CHANNELS :
  1   7   8  14  15  16  17  18  19  24  25  26  27  29

 SENSITIVITY AND GRADIENT ANALYSIS IS GIVEN BELOW:

 NORMALIZED GRADIENTS (I):

I=  1  II=  1 GN(I,II)=  1.000000
I=  2  II=  1 GN(I,II)=  0.559330
I=  3  II=  1 GN(I,II)=  0.549545
I=  4  II=  1 GN(I,II)=  0.503114
I=  5  II=  1 GN(I,II)=  0.239062
I=  6  II=  1 GN(I,II)=  0.216466
I=  7  II=  1 GN(I,II)=  0.406756
I=  8  II=  1 GN(I,II)=  0.411019
I=  9  II=  1 GN(I,II)=  0.744432
I= 10  II=  1 GN(I,II)=  0.655600
I= 11  II=  1 GN(I,II)=  0.700409
I= 12  II=  1 GN(I,II)=  0.203191
   1          I=  1  II=  1 GN(I,II)=  1.000000
   2          I=  9  II=  1 GN(I,II)=  0.744432
   3          I= 11  II=  1 GN(I,II)=  0.700409
   4          I= 10  II=  1 GN(I,II)=  0.655600
   5          I=  2  II=  1 GN(I,II)=  0.559330
   6          I=  3  II=  1 GN(I,II)=  0.549545
   7          I=  4  II=  1 GN(I,II)=  0.503114
   8          I=  8  II=  1 GN(I,II)=  0.411019
   9          I=  7  II=  1 GN(I,II)=  0.406756
  10          I=  5  II=  1 GN(I,II)=  0.239062
  11          I=  6  II=  1 GN(I,II)=  0.216466
  12          I= 12  II=  1 GN(I,II)=  0.203191


I=  1  II=  2 GN(I,II)=  1.000000
I=  2  II=  2 GN(I,II)=  0.558878
I=  3  II=  2 GN(I,II)=  0.579993
I=  4  II=  2 GN(I,II)=  0.558160
I=  5  II=  2 GN(I,II)=  0.304228
I=  6  II=  2 GN(I,II)=  0.166716
I=  7  II=  2 GN(I,II)=  0.544635
I=  8  II=  2 GN(I,II)=  0.556840
I=  9  II=  2 GN(I,II)=  0.645087
I= 10  II=  2 GN(I,II)=  0.606785
I= 11  II=  2 GN(I,II)=  0.591597
I= 12  II=  2 GN(I,II)=  0.154227
   1          I=  1  II=  2 GN(I,II)=  1.000000
   2          I=  9  II=  2 GN(I,II)=  0.645087
   3          I= 10  II=  2 GN(I,II)=  0.606785
   4          I= 11  II=  2 GN(I,II)=  0.591597
   5          I=  3  II=  2 GN(I,II)=  0.579993
   6          I=  2  II=  2 GN(I,II)=  0.558878
   7          I=  4  II=  2 GN(I,II)=  0.558160
   8          I=  8  II=  2 GN(I,II)=  0.556840
```

```
    9          I=  7  II=  2 GN(I,II)=  0.544635
   10          I=  5  II=  2 GN(I,II)=  0.304228
   11          I=  6  II=  2 GN(I,II)=  0.166716
   12          I= 12  II=  2 GN(I,II)=  0.154227


I=  1  II=  3 GN(I,II)=  1.000000
I=  2  II=  3 GN(I,II)=  0.479889
I=  3  II=  3 GN(I,II)=  0.583427
I=  4  II=  3 GN(I,II)=  0.389919
I=  5  II=  3 GN(I,II)=  0.173794
I=  6  II=  3 GN(I,II)=  0.214006
I=  7  II=  3 GN(I,II)=  0.539109
I=  8  II=  3 GN(I,II)=  0.647732
I=  9  II=  3 GN(I,II)=  0.675718
I= 10  II=  3 GN(I,II)=  0.575108
I= 11  II=  3 GN(I,II)=  0.563166
I= 12  II=  3 GN(I,II)=  0.232982
    1          I=  1  II=  3 GN(I,II)=  1.000000
    2          I=  9  II=  3 GN(I,II)=  0.675718
    3          I=  8  II=  3 GN(I,II)=  0.647732
    4          I=  3  II=  3 GN(I,II)=  0.583427
    5          I= 10  II=  3 GN(I,II)=  0.575108
    6          I= 11  II=  3 GN(I,II)=  0.563166
    7          I=  7  II=  3 GN(I,II)=  0.539109
    8          I=  2  II=  3 GN(I,II)=  0.479889
    9          I=  4  II=  3 GN(I,II)=  0.389919
   10          I= 12  II=  3 GN(I,II)=  0.232982
   11          I=  6  II=  3 GN(I,II)=  0.214006
   12          I=  5  II=  3 GN(I,II)=  0.173794


I=  1  II=  4 GN(I,II)=  1.000000
I=  2  II=  4 GN(I,II)=  0.655980
I=  3  II=  4 GN(I,II)=  0.413416
I=  4  II=  4 GN(I,II)=  0.502437
I=  5  II=  4 GN(I,II)=  0.315819
I=  6  II=  4 GN(I,II)=  0.159500
I=  7  II=  4 GN(I,II)=  0.287140
I=  8  II=  4 GN(I,II)=  0.328293
I=  9  II=  4 GN(I,II)=  0.757330
I= 10  II=  4 GN(I,II)=  0.668108
I= 11  II=  4 GN(I,II)=  0.735923
I= 12  II=  4 GN(I,II)=  0.129023
    1          I=  1  II=  4 GN(I,II)=  1.000000
    2          I=  9  II=  4 GN(I,II)=  0.757330
    3          I= 11  II=  4 GN(I,II)=  0.735923
    4          I= 10  II=  4 GN(I,II)=  0.668108
    5          I=  2  II=  4 GN(I,II)=  0.655980
    6          I=  4  II=  4 GN(I,II)=  0.502437
    7          I=  3  II=  4 GN(I,II)=  0.413416
    8          I=  8  II=  4 GN(I,II)=  0.328293
    9          I=  5  II=  4 GN(I,II)=  0.315819
   10          I=  7  II=  4 GN(I,II)=  0.287140
   11          I=  6  II=  4 GN(I,II)=  0.159500
   12          I= 12  II=  4 GN(I,II)=  0.129023


I=  1  II=  5 GN(I,II)=  1.000000
I=  2  II=  5 GN(I,II)=  0.676333
I=  3  II=  5 GN(I,II)=  0.313794
```

```
I=  4  II=  5 GN(I,II)=  0.404334
I=  5  II=  5 GN(I,II)=  0.369935
I=  6  II=  5 GN(I,II)=  0.131767
I=  7  II=  5 GN(I,II)=  0.325814
I=  8  II=  5 GN(I,II)=  0.292315
I=  9  II=  5 GN(I,II)=  0.558413
I= 10  II=  5 GN(I,II)=  0.618053
I= 11  II=  5 GN(I,II)=  0.615105
I= 12  II=  5 GN(I,II)=  0.148350
   1          I=  1  II=  5 GN(I,II)=  1.000000
   2          I=  2  II=  5 GN(I,II)=  0.676333
   3          I= 10  II=  5 GN(I,II)=  0.618053
   4          I= 11  II=  5 GN(I,II)=  0.615105
   5          I=  9  II=  5 GN(I,II)=  0.558413
   6          I=  4  II=  5 GN(I,II)=  0.404334
   7          I=  5  II=  5 GN(I,II)=  0.369935
   8          I=  7  II=  5 GN(I,II)=  0.325814
   9          I=  3  II=  5 GN(I,II)=  0.313794
  10          I=  8  II=  5 GN(I,II)=  0.292315
  11          I= 12  II=  5 GN(I,II)=  0.148350
  12          I=  6  II=  5 GN(I,II)=  0.131767


I=  1  II=  6 GN(I,II)=  1.000000
I=  2  II=  6 GN(I,II)=  0.474957
I=  3  II=  6 GN(I,II)=  0.518118
I=  4  II=  6 GN(I,II)=  0.394289
I=  5  II=  6 GN(I,II)=  0.190957
I=  6  II=  6 GN(I,II)=  0.206524
I=  7  II=  6 GN(I,II)=  0.380791
I=  8  II=  6 GN(I,II)=  0.431752
I=  9  II=  6 GN(I,II)=  0.664034
I= 10  II=  6 GN(I,II)=  0.637418
I= 11  II=  6 GN(I,II)=  0.621722
I= 12  II=  6 GN(I,II)=  0.206886
   1          I=  1  II=  6 GN(I,II)=  1.000000
   2          I=  9  II=  6 GN(I,II)=  0.664034
   3          I= 10  II=  6 GN(I,II)=  0.637418
   4          I= 11  II=  6 GN(I,II)=  0.621722
   5          I=  3  II=  6 GN(I,II)=  0.518118
   6          I=  2  II=  6 GN(I,II)=  0.474957
   7          I=  8  II=  6 GN(I,II)=  0.431752
   8          I=  4  II=  6 GN(I,II)=  0.394289
   9          I=  7  II=  6 GN(I,II)=  0.380791
  10          I= 12  II=  6 GN(I,II)=  0.206886
  11          I=  6  II=  6 GN(I,II)=  0.206524
  12          I=  5  II=  6 GN(I,II)=  0.190957


I=  1  II=  7 GN(I,II)=  1.000000
I=  2  II=  7 GN(I,II)=  0.500143
I=  3  II=  7 GN(I,II)=  0.518475
I=  4  II=  7 GN(I,II)=  0.413067
I=  5  II=  7 GN(I,II)=  0.202363
I=  6  II=  7 GN(I,II)=  0.202505
I=  7  II=  7 GN(I,II)=  0.379290
I=  8  II=  7 GN(I,II)=  0.422541
I=  9  II=  7 GN(I,II)=  0.680565
I= 10  II=  7 GN(I,II)=  0.640635
I= 11  II=  7 GN(I,II)=  0.640155
I= 12  II=  7 GN(I,II)=  0.200867
   1          I=  1  II=  7 GN(I,II)=  1.000000
```

```
   2          I=  9  II=  7 GN(I,II)=  0.680565
   3          I= 10  II=  7 GN(I,II)=  0.640635
   4          I= 11  II=  7 GN(I,II)=  0.640155
   5          I=  3  II=  7 GN(I,II)=  0.518475
   6          I=  2  II=  7 GN(I,II)=  0.500143
   7          I=  8  II=  7 GN(I,II)=  0.422541
   8          I=  4  II=  7 GN(I,II)=  0.413067
   9          I=  7  II=  7 GN(I,II)=  0.379290
  10          I=  6  II=  7 GN(I,II)=  0.202505
  11          I=  5  II=  7 GN(I,II)=  0.202363
  12          I= 12  II=  7 GN(I,II)=  0.200867


I=  1  II=  8 GN(I,II)=  1.000000
I=  2  II=  8 GN(I,II)=  0.490599
I=  3  II=  8 GN(I,II)=  0.554751
I=  4  II=  8 GN(I,II)=  0.446676
I=  5  II=  8 GN(I,II)=  0.200770
I=  6  II=  8 GN(I,II)=  0.240991
I=  7  II=  8 GN(I,II)=  0.502142
I=  8  II=  8 GN(I,II)=  0.648075
I=  9  II=  8 GN(I,II)=  0.752592
I= 10  II=  8 GN(I,II)=  0.601026
I= 11  II=  8 GN(I,II)=  0.616311
I= 12  II=  8 GN(I,II)=  0.229804
   1          I=  1  II=  8 GN(I,II)=  1.000000
   2          I=  9  II=  8 GN(I,II)=  0.752592
   3          I=  8  II=  8 GN(I,II)=  0.648075
   4          I= 11  II=  8 GN(I,II)=  0.616311
   5          I= 10  II=  8 GN(I,II)=  0.601026
   6          I=  3  II=  8 GN(I,II)=  0.554751
   7          I=  7  II=  8 GN(I,II)=  0.502142
   8          I=  2  II=  8 GN(I,II)=  0.490599
   9          I=  4  II=  8 GN(I,II)=  0.446676
  10          I=  6  II=  8 GN(I,II)=  0.240991
  11          I= 12  II=  8 GN(I,II)=  0.229804
  12          I=  5  II=  8 GN(I,II)=  0.200770


I=  1  II=  9 GN(I,II)=  1.000000
I=  2  II=  9 GN(I,II)=  0.668687
I=  3  II=  9 GN(I,II)=  0.725294
I=  4  II=  9 GN(I,II)=  0.659393
I=  5  II=  9 GN(I,II)=  0.252636
I=  6  II=  9 GN(I,II)=  0.187239
I=  7  II=  9 GN(I,II)=  0.581554
I=  8  II=  9 GN(I,II)=  0.601581
I=  9  II=  9 GN(I,II)=  0.894641
I= 10  II=  9 GN(I,II)=  0.626010
I= 11  II=  9 GN(I,II)=  0.759446
I= 12  II=  9 GN(I,II)=  0.165258
   1          I=  1  II=  9 GN(I,II)=  1.000000
   2          I=  9  II=  9 GN(I,II)=  0.894641
   3          I= 11  II=  9 GN(I,II)=  0.759446
   4          I=  3  II=  9 GN(I,II)=  0.725294
   5          I=  2  II=  9 GN(I,II)=  0.668687
   6          I=  4  II=  9 GN(I,II)=  0.659393
   7          I= 10  II=  9 GN(I,II)=  0.626010
   8          I=  8  II=  9 GN(I,II)=  0.601581
   9          I=  7  II=  9 GN(I,II)=  0.581554
  10          I=  5  II=  9 GN(I,II)=  0.252636
  11          I=  6  II=  9 GN(I,II)=  0.187239
```

```
  12          I= 12  II=  9 GN(I,II)=  0.165258


I=  1  II= 10 GN(I,II)=  1.000000
I=  2  II= 10 GN(I,II)=  0.515815
I=  3  II= 10 GN(I,II)=  0.541758
I=  4  II= 10 GN(I,II)=  0.429841
I=  5  II= 10 GN(I,II)=  0.203746
I=  6  II= 10 GN(I,II)=  0.173021
I=  7  II= 10 GN(I,II)=  0.360542
I=  8  II= 10 GN(I,II)=  0.380005
I=  9  II= 10 GN(I,II)=  0.681434
I= 10  II= 10 GN(I,II)=  0.650316
I= 11  II= 10 GN(I,II)=  0.653517
I= 12  II= 10 GN(I,II)=  0.169330
   1          I=  1  II= 10 GN(I,II)=  1.000000
   2          I=  9  II= 10 GN(I,II)=  0.681434
   3          I= 11  II= 10 GN(I,II)=  0.653517
   4          I= 10  II= 10 GN(I,II)=  0.650316
   5          I=  3  II= 10 GN(I,II)=  0.541758
   6          I=  2  II= 10 GN(I,II)=  0.515815
   7          I=  4  II= 10 GN(I,II)=  0.429841
   8          I=  8  II= 10 GN(I,II)=  0.380005
   9          I=  7  II= 10 GN(I,II)=  0.360542
  10          I=  5  II= 10 GN(I,II)=  0.203746
  11          I=  6  II= 10 GN(I,II)=  0.173021
  12          I= 12  II= 10 GN(I,II)=  0.169330


I=  1  II= 11 GN(I,II)=  1.000000
I=  2  II= 11 GN(I,II)=  0.536589
I=  3  II= 11 GN(I,II)=  0.541593
I=  4  II= 11 GN(I,II)=  0.428579
I=  5  II= 11 GN(I,II)=  0.201385
I=  6  II= 11 GN(I,II)=  0.169133
I=  7  II= 11 GN(I,II)=  0.349946
I=  8  II= 11 GN(I,II)=  0.363721
I=  9  II= 11 GN(I,II)=  0.697665
I= 10  II= 11 GN(I,II)=  0.652543
I= 11  II= 11 GN(I,II)=  0.671761
I= 12  II= 11 GN(I,II)=  0.167954
   1          I=  1  II= 11 GN(I,II)=  1.000000
   2          I=  9  II= 11 GN(I,II)=  0.697665
   3          I= 11  II= 11 GN(I,II)=  0.671761
   4          I= 10  II= 11 GN(I,II)=  0.652543
   5          I=  3  II= 11 GN(I,II)=  0.541593
   6          I=  2  II= 11 GN(I,II)=  0.536589
   7          I=  4  II= 11 GN(I,II)=  0.428579
   8          I=  8  II= 11 GN(I,II)=  0.363721
   9          I=  7  II= 11 GN(I,II)=  0.349946
  10          I=  5  II= 11 GN(I,II)=  0.201385
  11          I=  6  II= 11 GN(I,II)=  0.169133
  12          I= 12  II= 11 GN(I,II)=  0.167954


I=  1  II= 12 GN(I,II)=  1.000000
I=  2  II= 12 GN(I,II)=  0.471052
I=  3  II= 12 GN(I,II)=  0.511280
I=  4  II= 12 GN(I,II)=  0.376098
I=  5  II= 12 GN(I,II)=  0.182079
I=  6  II= 12 GN(I,II)=  0.210430
```

```
I=  7  II= 12 GN(I,II)=  0.378250
I=  8  II= 12 GN(I,II)=  0.438089
I=  9  II= 12 GN(I,II)=  0.663775
I= 10  II= 12 GN(I,II)=  0.633581
I= 11  II= 12 GN(I,II)=  0.618131
I= 12  II= 12 GN(I,II)=  0.214441
   1          I=  1  II= 12 GN(I,II)=  1.000000
   2          I=  9  II= 12 GN(I,II)=  0.663775
   3          I= 10  II= 12 GN(I,II)=  0.633581
   4          I= 11  II= 12 GN(I,II)=  0.618131
   5          I=  3  II= 12 GN(I,II)=  0.511280
   6          I=  2  II= 12 GN(I,II)=  0.471052
   7          I=  8  II= 12 GN(I,II)=  0.438089
   8          I=  7  II= 12 GN(I,II)=  0.378250
   9          I=  4  II= 12 GN(I,II)=  0.376098
  10          I= 12  II= 12 GN(I,II)=  0.214441
  11          I=  6  II= 12 GN(I,II)=  0.210430
  12          I=  5  II= 12 GN(I,II)=  0.182079


 SENS END

 GRAD START:
  0      -0.6614E-01  0.3469E-01 -0.6375E-01 -0.2349E+00 -0.1611E+00
  5      -0.1783E-01 -0.2103E-01  0.7394E-01 -0.7075E-01 -0.6224E-01
 10      -0.9647E-01 -0.8617E-01  0.4081E-02 -0.1605E-02  0.2710E-02
 15       0.1334E-01  0.9411E-02  0.3488E-03  0.8920E-03 -0.3658E-02
 20       0.3762E-02  0.3701E-02  0.5824E-02  0.4168E-02  0.5274E-01
 25      -0.2719E-01  0.4108E-01  0.1798E+00  0.1201E+00  0.1143E-01
 30       0.2269E-01 -0.5457E-01  0.5146E-01  0.4551E-01  0.7547E-01
 35       0.6029E-01  0.3284E-01 -0.1811E-01  0.2413E-01  0.1239E+00
 40       0.9032E-01  0.1099E-01  0.1299E-01 -0.3642E-01  0.2536E-01
 45       0.3287E-01  0.4667E-01  0.4344E-01  0.2378E-01 -0.1128E-01
 50       0.2962E-01  0.8830E-01  0.6289E-01  0.6424E-02  0.2667E-02
 55      -0.2902E-01  0.3085E-01  0.2545E-01  0.3626E-01  0.3442E-01
 60       0.3689E-01 -0.1849E-01  0.2417E-01  0.1211E+00  0.8715E-01
 65       0.1212E-01  0.9193E-02 -0.3602E-01  0.3084E-01  0.3407E-01
 70       0.4775E-01  0.4257E-01  0.8051E-02 -0.3961E-02  0.5648E-02
 75       0.2576E-01  0.1730E-01  0.1965E-02  0.3006E-02 -0.7777E-02
 80       0.7474E-02  0.6821E-02  0.1084E-01  0.8613E-02  0.3381E-01
 85      -0.1597E-01  0.3498E-01  0.1128E+00  0.7547E-01  0.7111E-02
 90       0.7041E-02 -0.3648E-01  0.4214E-01  0.3082E-01  0.4878E-01
 95       0.4148E-01  0.2747E-01 -0.1365E-01  0.2968E-01  0.1009E+00
100       0.7072E-01  0.7284E-02  0.7014E-02 -0.3210E-01  0.3222E-01
105       0.2786E-01  0.4126E-01  0.3755E-01  0.3968E-01 -0.2721E+00
110      -0.8928E+00  0.4557E+00 -0.2129E+00  0.7342E-01  0.1293E+00
115       0.1667E+00 -0.8126E-02  0.8819E-02  0.8776E-02 -0.2694E-02
120      -0.2145E-02 -0.7255E-02 -0.7578E-02  0.8306E-02  0.8088E-02
125      -0.7855E-02 -0.7892E-02 -0.8761E-02  0.1539E+00 -0.1535E+00
130      -0.1525E+00  0.8009E-01  0.8122E-01  0.1516E+00  0.1553E+00
135      -0.1547E+00 -0.1486E+00  0.1731E+00  0.1716E+00  0.1679E+00
140       0.4667E+00 -0.4623E+00 -0.4588E+00  0.2373E+00  0.2392E+00
145       0.4540E+00  0.4642E+00 -0.4633E+00 -0.4448E+00  0.5124E+00
150       0.5095E+00  0.5050E+00 -0.2709E+00  0.2690E+00  0.2679E+00
155      -0.1441E+00 -0.1456E+00 -0.2682E+00 -0.2743E+00  0.2709E+00
160       0.2607E+00 -0.3051E+00 -0.3050E+00 -0.2972E+00  0.1408E+00
165      -0.1390E+00 -0.1381E+00  0.7500E-01  0.7661E-01  0.1385E+00
170       0.1416E+00 -0.1405E+00 -0.1348E+00  0.1573E+00  0.1557E+00
175       0.1533E+00 -0.6713E-02  0.7706E-02  0.6728E-02  0.8169E-03
180       0.1260E-02 -0.4530E-02 -0.4437E-02  0.6640E-02  0.5801E-02
185      -0.4263E-02 -0.4053E-02 -0.5261E-02 -0.9481E-01  0.9320E-01
190       0.9262E-01 -0.4948E-01 -0.5104E-01 -0.9169E-01 -0.9446E-01
195       0.9434E-01  0.9039E-01 -0.1044E+00 -0.1012E+00 -0.1016E+00
```

```
200       -0.1063E+00  0.1015E+00  0.1009E+00 -0.5247E-01 -0.5420E-01
205       -0.9919E-01 -0.1019E+00  0.1019E+00  0.9771E-01 -0.1106E+00
210       -0.1075E+00 -0.1104E+00  0.0000E+00  0.0000E+00  0.0000E+00


output layer biases:

K,GG,=            1   -6.614041E-02
K,GG,=            2    3.469139E-02
K,GG,=            3   -6.374832E-02
K,GG,=            4   -0.234920
K,GG,=            5   -0.161140
K,GG,=            6   -1.783263E-02
K,GG,=            7   -2.102808E-02
K,GG,=            8    7.393961E-02
K,GG,=            9   -7.074858E-02
K,GG,=           10   -6.223520E-02
K,GG,=           11   -9.647085E-02
K,GG,=           12   -8.616605E-02
 hidden-to-output layer weights:

K,GG=            13    4.080878E-03
K,GG=            14   -1.604972E-03
K,GG=            15    2.710276E-03
K,GG=            16    1.334336E-02
K,GG=            17    9.410932E-03
K,GG=            18    3.487800E-04
K,GG=            19    8.920036E-04
K,GG=            20   -3.658458E-03
K,GG=            21    3.762130E-03
K,GG=            22    3.701499E-03
K,GG=            23    5.824398E-03
K,GG=            24    4.167868E-03
K,GG=            25    5.273741E-02
K,GG=            26   -2.719065E-02
K,GG=            27    4.108173E-02
K,GG=            28    0.179829
K,GG=            29    0.120089
K,GG=            30    1.143054E-02
K,GG=            31    2.269197E-02
K,GG=            32   -5.457142E-02
K,GG=            33    5.145916E-02
K,GG=            34    4.551226E-02
K,GG=            35    7.547205E-02
K,GG=            36    6.029389E-02
K,GG=            37    3.284172E-02
K,GG=            38   -1.811421E-02
K,GG=            39    2.413063E-02
K,GG=            40    0.123851
K,GG=            41    9.031557E-02
K,GG=            42    1.098555E-02
K,GG=            43    1.299263E-02
K,GG=            44   -3.642125E-02
K,GG=            45    2.536314E-02
K,GG=            46    3.287464E-02
K,GG=            47    4.666984E-02
K,GG=            48    4.344149E-02
K,GG=            49    2.378142E-02
K,GG=            50   -1.127665E-02
K,GG=            51    2.961641E-02
K,GG=            52    8.829614E-02
K,GG=            53    6.289395E-02
K,GG=            54    6.423504E-03
```

```
K,GG=          55    2.667449E-03
K,GG=          56   -2.902022E-02
K,GG=          57    3.085108E-02
K,GG=          58    2.544772E-02
K,GG=          59    3.625873E-02
K,GG=          60    3.441710E-02
K,GG=          61    3.689079E-02
K,GG=          62   -1.848991E-02
K,GG=          63    2.417157E-02
K,GG=          64    0.121129
K,GG=          65    8.714787E-02
K,GG=          66    1.211667E-02
K,GG=          67    9.193145E-03
K,GG=          68   -3.601532E-02
K,GG=          69    3.084007E-02
K,GG=          70    3.406567E-02
K,GG=          71    4.774650E-02
K,GG=          72    4.257489E-02
K,GG=          73    8.050930E-03
K,GG=          74   -3.960771E-03
K,GG=          75    5.647793E-03
K,GG=          76    2.575507E-02
K,GG=          77    1.730228E-02
K,GG=          78    1.965344E-03
K,GG=          79    3.005559E-03
K,GG=          80   -7.777176E-03
K,GG=          81    7.474252E-03
K,GG=          82    6.820793E-03
K,GG=          83    1.083673E-02
K,GG=          84    8.613389E-03
K,GG=          85    3.381192E-02
K,GG=          86   -1.597272E-02
K,GG=          87    3.498291E-02
K,GG=          88    0.112808
K,GG=          89    7.546914E-02
K,GG=          90    7.111365E-03
K,GG=          91    7.041377E-03
K,GG=          92   -3.647536E-02
K,GG=          93    4.214221E-02
K,GG=          94    3.081526E-02
K,GG=          95    4.877988E-02
K,GG=          96    4.147670E-02
K,GG=          97    2.747182E-02
K,GG=          98   -1.364729E-02
K,GG=          99    2.967993E-02
K,GG=         100    0.100854
K,GG=         101    7.071928E-02
K,GG=         102    7.283851E-03
K,GG=         103    7.013907E-03
K,GG=         104   -3.210177E-02
K,GG=         105    3.222409E-02
K,GG=         106    2.786051E-02
K,GG=         107    4.126469E-02
K,GG=         108    3.755478E-02

EXTRA HIDDEN LAYER BIASES (1ST FROM END:
K,GG=         109    3.967559E-02
K,GG=         110   -0.272070
K,GG=         111   -0.892778
K,GG=         112    0.455690
K,GG=         113   -0.212906
K,GG=         114    7.341579E-02
K,GG=         115    0.129310
K,GG=         116    0.166746
```

```
input-to-hidden layer weights:
 (( J=1,L), I=1,M) ; L,M=          12           8
K,GG=          117   -8.125763E-03
K,GG=          118    8.818868E-03
K,GG=          119    8.776159E-03
K,GG=          120   -2.694123E-03
K,GG=          121   -2.145039E-03
K,GG=          122   -7.255481E-03
K,GG=          123   -7.578345E-03
K,GG=          124    8.305606E-03
K,GG=          125    8.088467E-03
K,GG=          126   -7.854603E-03
K,GG=          127   -7.891879E-03
K,GG=          128   -8.760776E-03
K,GG=          129    0.153939
K,GG=          130   -0.153538
K,GG=          131   -0.152495
K,GG=          132    8.008923E-02
K,GG=          133    8.121772E-02
K,GG=          134    0.151624
K,GG=          135    0.155300
K,GG=          136   -0.154661
K,GG=          137   -0.148599
K,GG=          138    0.173109
K,GG=          139    0.171623
K,GG=          140    0.167946
K,GG=          141    0.466669
K,GG=          142   -0.462262
K,GG=          143   -0.458845
K,GG=          144    0.237291
K,GG=          145    0.239220
K,GG=          146    0.454046
K,GG=          147    0.464225
K,GG=          148   -0.463275
K,GG=          149   -0.444755
K,GG=          150    0.512448
K,GG=          151    0.509535
K,GG=          152    0.505020
K,GG=          153   -0.270913
K,GG=          154    0.269021
K,GG=          155    0.267898
K,GG=          156   -0.144053
K,GG=          157   -0.145550
K,GG=          158   -0.268246
K,GG=          159   -0.274305
K,GG=          160    0.270936
K,GG=          161    0.260693
K,GG=          162   -0.305089
K,GG=          163   -0.304982
K,GG=          164   -0.297224
K,GG=          165    0.140780
K,GG=          166   -0.138985
K,GG=          167   -0.138128
K,GG=          168    7.499578E-02
K,GG=          169    7.660617E-02
K,GG=          170    0.138538
K,GG=          171    0.141574
K,GG=          172   -0.140466
K,GG=          173   -0.134821
K,GG=          174    0.157322
K,GG=          175    0.155703
K,GG=          176    0.153272
```

```
K,GG=        177   -6.712943E-03
K,GG=        178    7.705803E-03
K,GG=        179    6.727723E-03
K,GG=        180    8.169123E-04
K,GG=        181    1.260246E-03
K,GG=        182   -4.530146E-03
K,GG=        183   -4.437063E-03
K,GG=        184    6.640156E-03
K,GG=        185    5.800942E-03
K,GG=        186   -4.262775E-03
K,GG=        187   -4.053425E-03
K,GG=        188   -5.260953E-03
K,GG=        189   -9.480628E-02
K,GG=        190    9.319815E-02
K,GG=        191    9.262326E-02
K,GG=        192   -4.947815E-02
K,GG=        193   -5.104069E-02
K,GG=        194   -9.168660E-02
K,GG=        195   -9.445944E-02
K,GG=        196    9.434228E-02
K,GG=        197    9.038814E-02
K,GG=        198   -0.104426
K,GG=        199   -0.101160
K,GG=        200   -0.101649
K,GG=        201   -0.106335
K,GG=        202    0.101508
K,GG=        203    0.100948
K,GG=        204   -5.247010E-02
K,GG=        205   -5.419755E-02
K,GG=        206   -9.918871E-02
K,GG=        207   -0.101857
K,GG=        208    0.101924
K,GG=        209    9.770510E-02
K,GG=        210   -0.110572
K,GG=        211   -0.107483
K,GG=        212   -0.110412
```

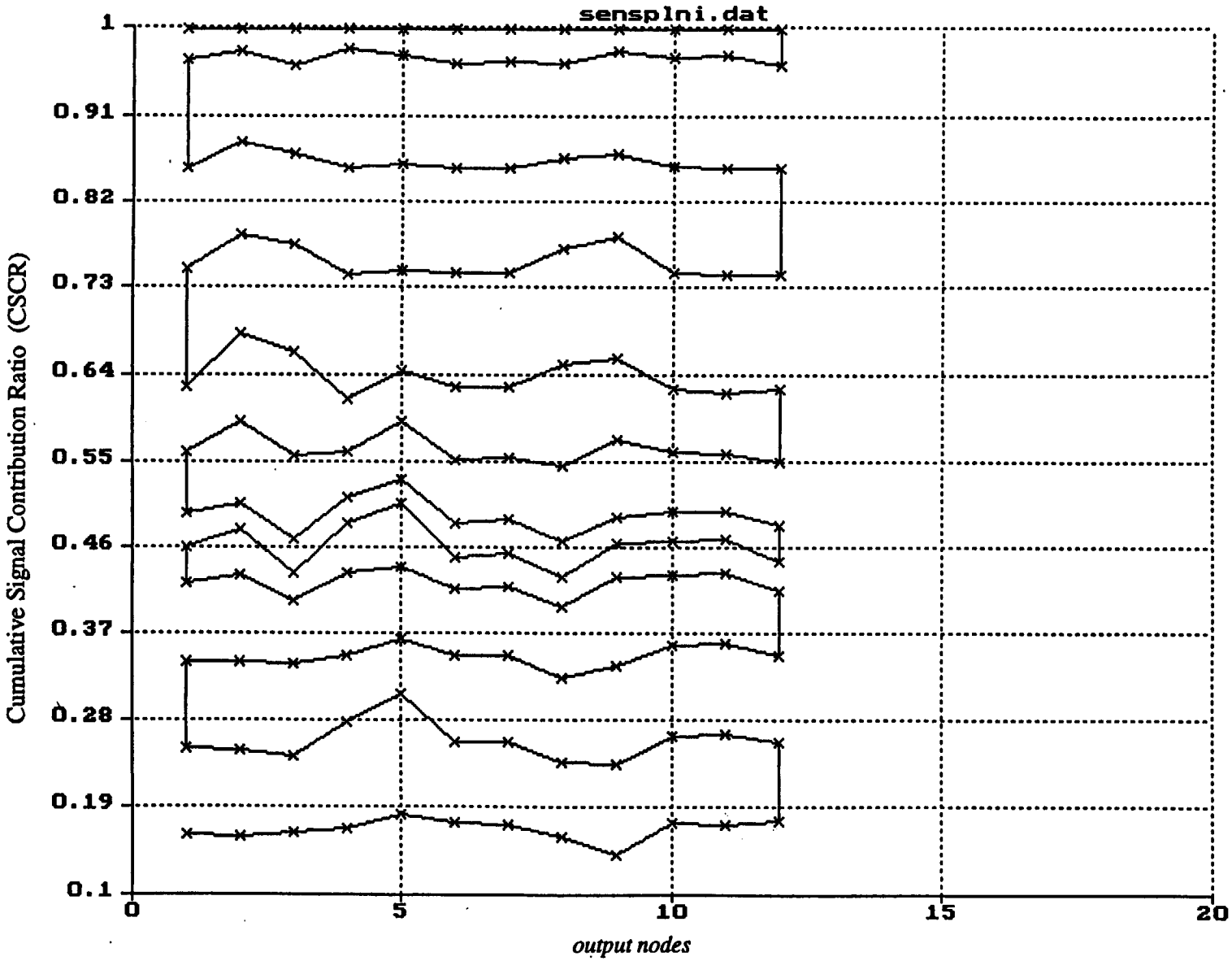

Fig.56 : Sensitivity analysis with additive noise. The horizontal-axis represents the output nodes. Each relative contribution to each output node is represented on the vertical-axis. The conributions are in cumulative form so that it amounts to unity for all signals. The network is autoassociative and the learning algorithm is variable-metric.

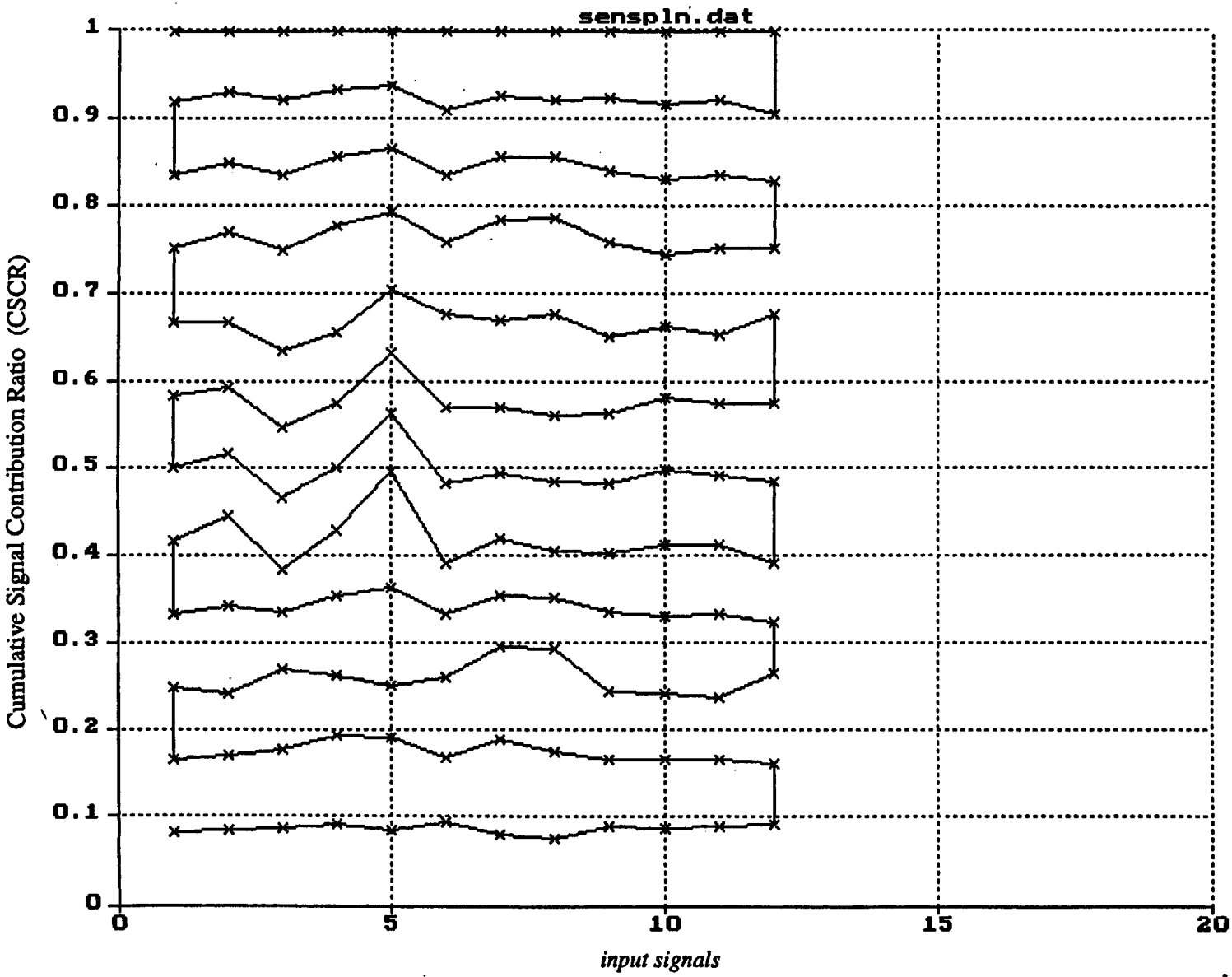

Fig.57 : Sensitivity analysis with additive noise. The horizontal-axis represents the input signals. Relative contribution of each input signal to all output nodes is represented on the vertical-axis. The conributions are in cumulative form so that it amounts to unity for all signals. The network is autoassociative and the learning algorithm is variable-metric.

REFERENCES

1. E. Turkcan et al., *Information rocessing system and neural network utilization for accident management support*, Proc. FISA-95 Symposium, EU Research on Severe Accidents, Luxemboutg, 20-22 November 95. Also appeared in Report, Netherland Energy Research Foundation, ECN-RX--94-031, May 1994, The Netherlands

2. E.Turkcan, O. Ciftcioglu and K. Nabeshima, *Neural networks for real-time NPP monitoring*, Nuclear Europe Worldscan 11-12 November/December 1993

3. O. Ciftcioglu and E. Turkcan, On-line plant-wide monitoring using neural networks, 8th Power Plant Dynamics, Control and Testing Symposium, May 27-29, 1992, Knoxville, Tennessee, (USA)

4. O. Ciftcioglu and E. Turkcan, *Neural network training by parameter optimization approach*, Proc. International Conference on Artificial Intelligence ICANN'93, Amsterdam, September 13-16,1993, (The Netherlands)

5. O. Ciftcioglu and E. Turkcan, *Optimal training for neural networks applied to nuclear technology*, Neural Networks: Artificial Intelligence and Industrial Applications, Proc. 3rd Annual SNN Symposium of Neural Networks, 14-15 September,Nijmegen, (The Netherlands), Eds. Bert Kappen and Stan Gielen, Springer Verlag ISBN 3-540-19992-6

6. O. Ciftcioglu and E. Turkcan, *Adaptive training of feedforward neural networks by Kalman filtering*, Report, Netherland Energy Research Foundation, ECN-R--95-001, February 1995, (The Netherlands)

7. OECD-NEA/NSC/DOC(96)17, May 1996, *OVERVIEW*, 7th Symposium on Reactor Surveillance and Diagnostics (SMORN -VII), 19th to 23rd June 1995, Avignon (France)

8. E. Turkcan and O. Ciftcioglu, *Neural Network Benchmark for SMORN-VII*, (Extended Summary Report), Proc. 7th Symposium on Reactor Surveillance and Diagnostics (SMORN-VII), 19-24 June, 1995, Avignon (France)

9. O. Ciftcioglu and E. Turkcan, *Wavelet Understands Neural Networks*, Proc. IEEE 2nd Signal Processing Symposium, 8-9 April 1994, Gokova, Turkey. Also appeared in Report, Netherlands Energy Research Foundation, ECN-RX--94-031, May 1994, (The Netherlands)