LRP 384/89 August 1989

# THE STANDARD DEVIATION METHOD: DATA ANALYSIS BY CLASSICAL MEANS AND BY NEURAL NETWORKS

G. Bugmann, J.B. Lister and U. von Stockar

Paper presented at the

**International Conference on Recent Development in Statistical Data Analysis and Inference**
**Neuchâtel, Switzerland**
**August 21-24, 1989**

# The Standard Deviation Method : Data analysis by classical means and by neural networks

G. Bugmann, J.B. Lister* and U. von Stockar

Inst. de Génie Chimique, EPFL, CH-1015 Lausanne, Suisse
*Centre de Recherches en Physique des Plasmas, Association EURATOM-Confédération Suisse, EPFL, CH-1007 Lausanne

## ABSTRACT

The Standard Deviation Method is a method for determining particle size which can be used, for instance, to determine air-bubble sizes[1] in a fermentation bio-reactor. The transmission coefficient of an ultrasound beam through a gassy liquid is measured repetitively. Due to the displacements and random positions of the bubbles, the measurements show a scatter whose standard deviation is dependent on the bubble-size. The precise relationship between the measured standard deviation, the transmission and the particle size has been obtained from a set of computer-simulated data.

During the computer modelling, the size and transmission are specified in advance, and the standard deviation is calculated. For the evaluation of the experimental data, we need to know the relationship between the measured parameters (transmission coefficient and standard deviation) and the unknown parameter (particle size). In order to develop the mapping between these measured inputs and the unknown output we have examined two very different approaches :

- A classical series expansion fitting procedure.
- Training of a neural network.

The particle size corresponding to real-time experimental data is then correspondingly determined by :

- An iterative procedure applied to the best-fit series expansion.
- Direct input to the neural network.

In this paper we describe and compare the two approaches.

# I INTRODUCTION

This paper describes firstly a new particle size measuring method and secondly shows how the numerical properties of neural networks can be used to solve the rather general problem of generating non-explicit mapping functions.

A size measuring method, called the Standard Deviation Method, uses the statistical properties of the intensity of a beam (light, ultrasound, etc) crossing a random dispersion of moving opaque particles. As this method is new, it will be described in some detail. In this way, the problem which is later posed to the neural network will be clearly defined.

Computer simulations have been used to generate the corresponding output data results for given input data, in order to produce sets of inter-related data. Some of these data are subsequently considered to be measurable quantities, and some are considered to be physical parameters of the system, to be derived from the measurements. The act of computer simulation denies the possibility of a closed expression for the mapping to the physical quantities, which is a rather frequently encountered problem in the experimental sciences.

More formally, we define a measurement vector $\underline{M} \varepsilon R^m$ contained in an m-dimensional space, and a physical description vector $\underline{G} \varepsilon R^p$ contained in a p-dimensional space. We must derive a formal mapping $\underline{G}(\underline{M})$, given a limited number of example sets $\{\underline{G}_i,\underline{M}_i\}$ which are considered adequate to define the mapping.

Neural networks are often used for their logical and associative properties[2]. Here, their continuous computational capacities are used, although the learning procedure is similar ; examples of the mapping are shown to the network which learns to reproduce them. In this way, it builds up an internal representation of the mapping function, which can be used later on cases not contained in the examples set. The neural network has learned to interpolate between the discrete examples of the mapping. This mapping is only valid within a space populated by the examples, as in the case of regular

fitting. A conventional back-propagation network has been used for this application.

The layout of this paper is as follows. In section II we discuss the Standard Deviation Method as applicable to the measurement of the bubble size in a bio-reactor. The computer generated data contain all the information relating the three interdependent variables, beam transmission, its standard deviation and the bubble size.

Section III describes a conventional development of a mapping function relating bubble size to the measurable quantities, deriving a certain quality of the representation, defined by the spread of the relative errors of the fit. In section IV we describe the use of a neural network to provide this mapping, and evaluate the same representational quality description. Since the quality is at least as good as the classical method, we then reduce the neural network size to explore its limitations. In section V we discuss the relative merits, especially invested effort, of the two techniques.

## II THE STANDARD DEVIATION METHOD

The Standard Deviation Method has been developed to measure bubble sizes in air-liquid dispersions as found in aerated bio-reactors[1]. The Standard Deviation Method is, however, not restricted to this special use, but is a generally applicable size determination method. For simplicity, we shall consider a system comprising a flow of randomly dispersed monosized opaque spheres crossing a light beam (Fig.1).

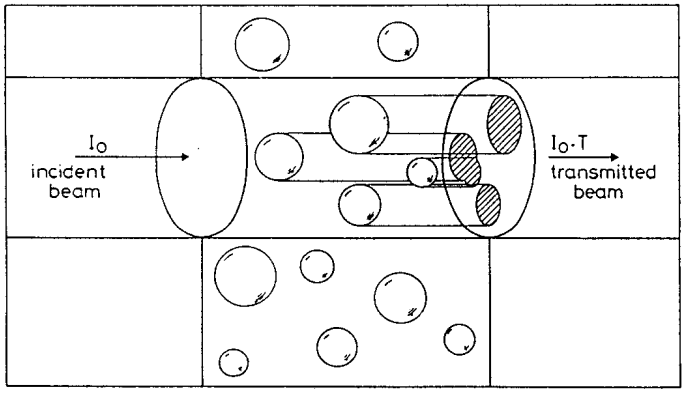


Figure 1: A schematic of the experimental method, showing the interception of the beam by the bubbles, reducing the transmission coefficient, T.

The spatially integrated intensity of the beam is reduced due to the presence of the spheres. A particular spheres configuration ($_i$) can be characterised by a transmission coefficient $T_i$ varying from 0 to 1. When there are no spheres in the beam, $T_i = 1$ and when the spheres are so densely packed that no light can get through, then $T_i = 0$. In intermediate real cases, repetitive measurements of $T_i$ produce values dispersed around a mean value T with a standard deviation $\sigma$ depending on the sphere size (Fig. 2).

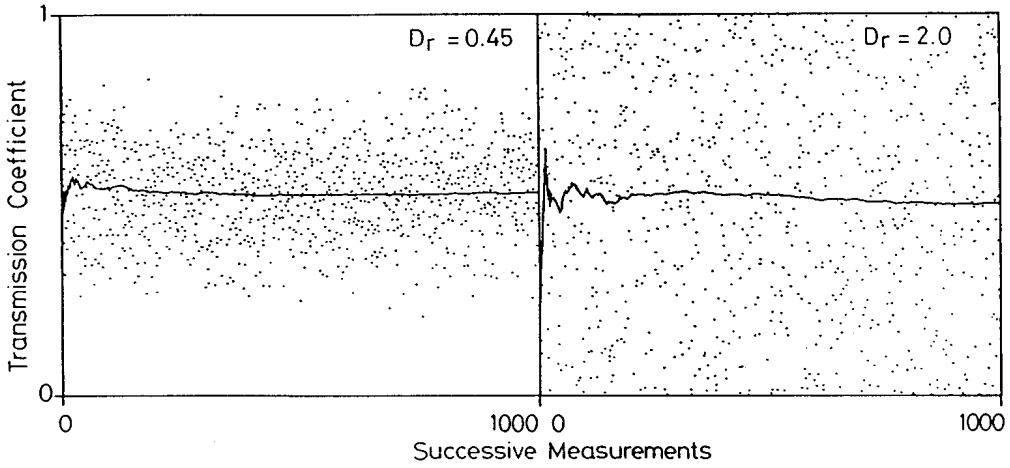


Figure 2: Successive simulation measurements of the transmission coefficient for two bubble sizes (Dr). The dispersion of the mean transmission is greater in the second case of larger bubbles. The solid line is the progressive average of the transmission coefficient.

To use $\sigma$ to determine the sphere diameter, it is necessary to know the relationship between $\sigma$, D and the density of the spheres which determines T. As such an analytical expression does not exist, computer simulations have been carried out to provide a set of $\sigma$ values for various combinations of D and T (Fig. 3). In this figure, the T dependence of $\sigma$ is visible. As the mean transmission value T approaches 1 or 0, $\sigma$ must also approach 0. This is due to the fact that when the mean T value is, for instance, 0 then no individual $T_i$ measurement can be larger than 0. Consequently the standard deviation $\sigma$ must also be 0. The values ($\sigma$, T) = (0, 0) and (1, 1) represent degenerate points for the mapping function, and are not used as examples for either of the functional mapping techniques.

In the Standard Deviation Method, the deviation of the measured points, which is usually considered to be a problem in a measurement system, is used as a source of information in itself.

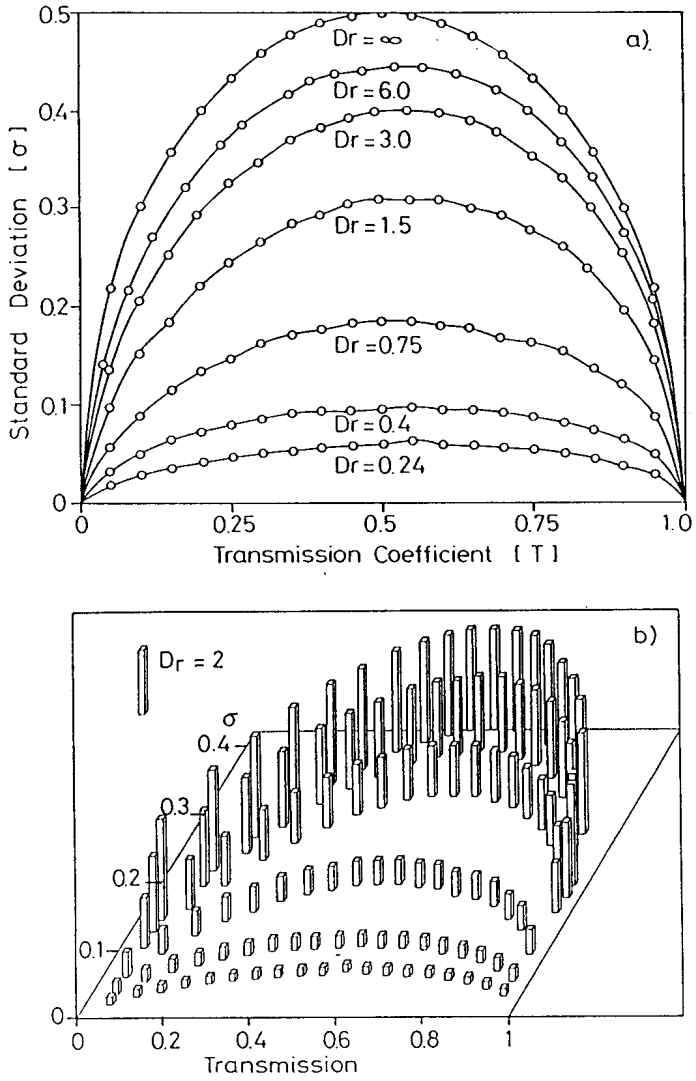


Figure 3: (a) The values of σ, T obtained for various simulations of different sizes, Dr. The mapping Dr (σ, T) is the challenge.

(b) An angled 3-D view of Fig. 3a), showing the amphitheatre form of the functional dependence. The block heights are the bubble sizes, Dr.

The curves of Fig. 3 can be applied to beams of any diameter $D_b$ since they only depend on the relative sphere diameter $Dr = D * 1.5/D_b$. $D_b/1.5$ is the mean diameter of the cylindrical beam. In what follows we shall consider Dr as the parameter to be derived.

The problem which is posed to the experimentalist is the following: given a measured value T and its corresponding σ, what is the value of Dr which best corresponds to these data? For instance, T=0.5 and σ=0.4 correspond to Dr=3.0. In order to provide a continous estimate of Dr, the algorithm must be able to interpolate between the documented example points.

## III CLASSICAL APPROACH

The experimentalist's problem would be very simple if there were an analytical expression $Dr = Dr(\sigma,T)$. The problem would still be simple if an analytical expression $\sigma = \sigma(T,Dr)$ were available. In that case, Dr could still be determined from T and $\sigma$ , by using an recursive procedure. In reality, there are no such expressions but it was possible to find an approximate expression for $\sigma = \sigma(T,Dr)$ by 2-dimensional fitting from the data corresponding to Fig. 3 .

The fitting was carried out in 3 steps :

First, all the curves of Fig. 3 were divided by $\sqrt{[T(T-0.5)]}$. This is the expression for $\sigma$ which is easily found in the extreme case of a beam of zero diameter. This division eliminates the problem of fitting the predominant bell-shape of all the curves.

Secondly, the result of this division was fitted, curve by curve, to the expression :

$$\Sigma [ a_k(Dr) * (T-0.5)^{k-1} ] \quad k=1 \text{ to } 4$$

The number of terms in the series was kept low in order to ensure a smooth fitting of the curves and a low computation time during subsequent evaluation.

Thirdly, the $a_k(Dr)$ parameters found for each curve were fitted to the expression :

$$a_k(Dr) = \Sigma [ B_{k,l} * Dr^l ] \qquad l=1 \text{ to } 6$$

This form allows a precise fitting for Dr values up to 3.0, i.e. for spheres of a diameter up to twice the beam diameter. In principle, it is unsatisfactory to have only a partial mapping of the $(\sigma,T)$ space, but, for practical use, this limitation is not very important. Finally, the full approximated expression for $\sigma$, using the 24 parameters $B_{k,l}$, becomes :

$$\sigma(T,Dr) = \sqrt{[T(T-0.5)]} * \Sigma \{(T-0.5)^{k-1} * \Sigma [ B_{k,l} * Dr^l ] \} \quad k=1 \text{ to } 4 \text{ ; } l=1 \text{ to } 6$$

In order to find the Dr corresponding to a given measurement couple $(\sigma,T)$, we have to make an initial estimate of Dr and use it to calculate $\sigma(T,Dr)$. The comparison between the given $\sigma$ and the calculated one gives us a better estimate of Dr, and so on. This iterative procedure allows us to determine Dr as precisely as necessary, within the accuracy of the fitted function.

To determine the accuracy of the σ(T,Dr) approximation which we have derived, the Dr values calculated from the (σ,T) data in Fig. 3 have been compared with the Dr values initially set in the simulation program which originally produced Fig. 3. The resulting accuracies are shown in Figs 4 and 5. The relative error is denoted by the size of the circles in Fig. 4a. A histogram of the relative errors for all the 114 fitted data points is given in Fig. 5a. The root mean square (RMS) of the relative error is 3 %. In the major part of the known (σ,T) space the accuracy is better than 5%, which is acceptable for a measuring instrument. One should remember that the {σ, T} values are the result of a Monte-Carlo type simulation and that their intrinsic accuracy is limited. The quality of representation found for this classical Dr(T,σ) mapping is the challenge put to a competing technique.

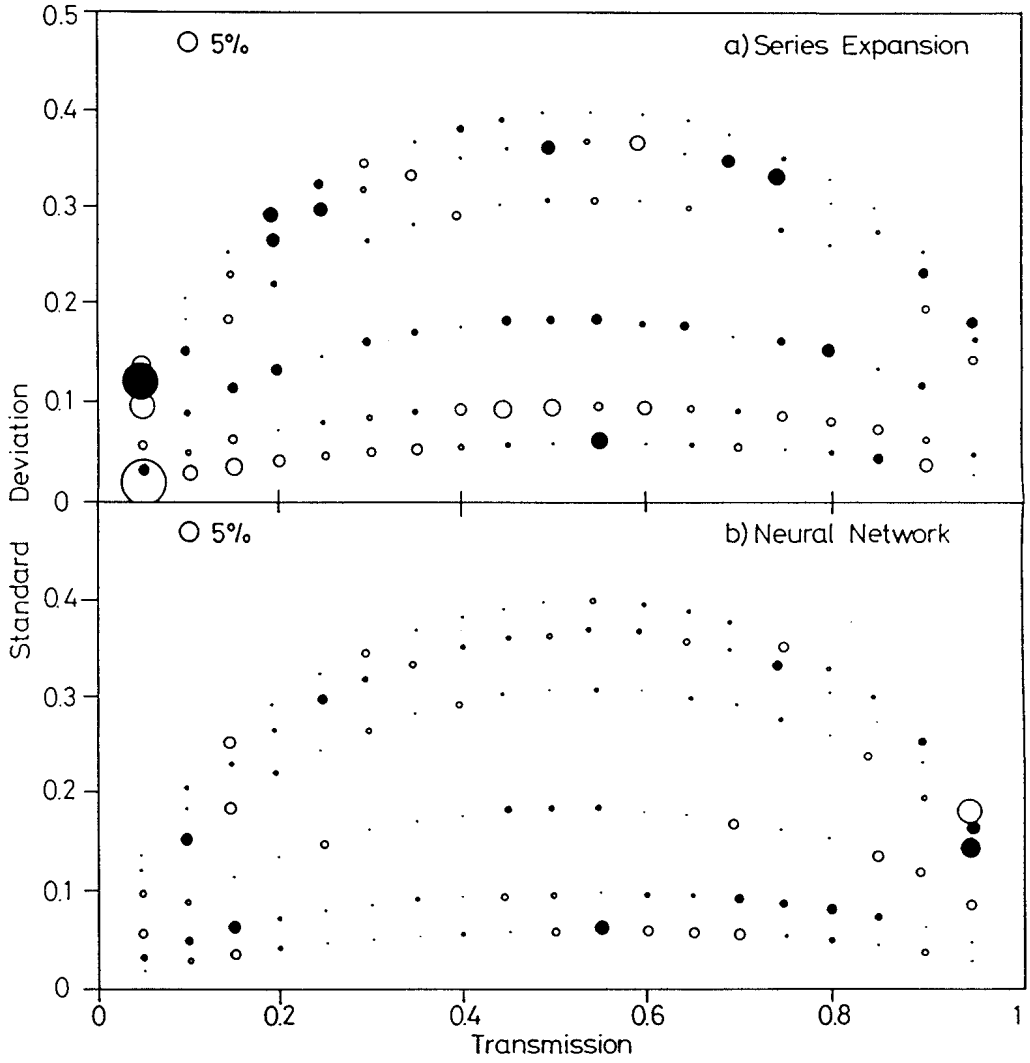


Figure 4: The errors (example value - fitted value) found with the two fitting techniques, represented by the diameter of the circles, filled circles being negative errors: a) Series Expansion, b) Neural Network.

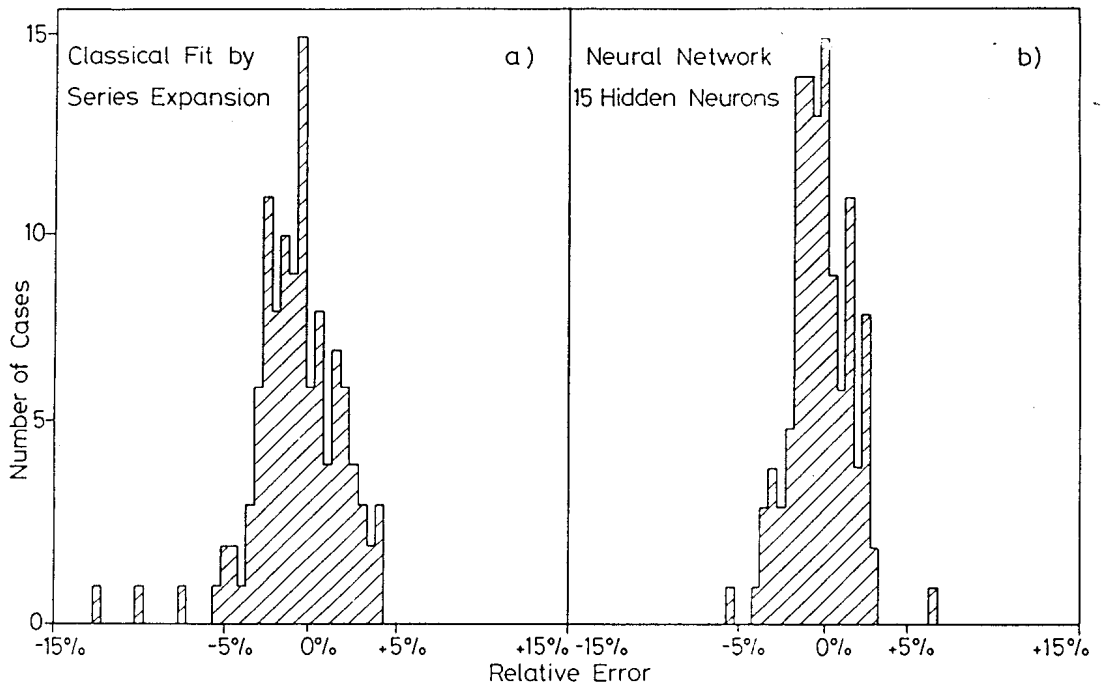


Figure 5: Histogram of the relative errors for a) Series Expansion and b) Neural Network.

## IV THE NEURAL NETWORK APPROACH

### IV.1 The network

This section describes the application of a neural network to yield directly the mapping $Dr(T,\sigma)$. Our aim was to use conventional neural network techniques with no attempt to adapt them at all, or to improve on them. We first give a brief resumé of the neural network to be used. More background details are found in the References (2,3).

Figure 6 shows the configuration of a general 3-layer, that is to say one hidden layer, network used in this study. In our case the input vector is the two variables $\{\sigma,T\}$, and is presented to the two external inputs. A third "input" fixed at 1.0 provides a bias term. Each of the inputs (j) is connected to each hidden neuron (i) by a weight $W_{ij}^{12}$ such that the sums of the connections to the hidden layer neurons are given by the matrix multiplication $\underline{\Sigma 2} = \mathbf{W}^{12} \cdot \underline{S1}$, where $\underline{S1}$ is the 3-element input vector, $\underline{\Sigma 2}$ is the hidden neuron layer input vector and $\mathbf{W}^{12}$ is the matrix of weights between layer 1 and layer 2.

The hidden layer neurons have an asymmetric non-linear sigmoidal transfer function $S2_i = (1 + \exp\text{-}\Sigma 2_i)^{-1}$. This non-linearity is vital in separating the two matrices $\mathbf{W}^{12}$ and $\mathbf{W}^{23}$ . If the transfer function were linear, the two matrices would become one single matrix by linear algebra, and the hidden neurons would have no sense. The significance of the bias term added to the input vector is also clear, as it determines the value of the other summed inputs at which the output is 0.5. A further bias neuron is added to this $\underline{S2}$ vector and the data transformation is repeated via a new matrix $\mathbf{W}^{23}$ linking the hidden layer output $\underline{S2}$ to the output layer input $\underline{\Sigma 3}$ : $\underline{\Sigma 3} = \mathbf{W}^{23} \cdot \underline{S2}$ . This output layer neuron has the same sigmoidal transfer function, yielding the final network output vector $\underline{S3}$ , in our simple case the single element Dr.

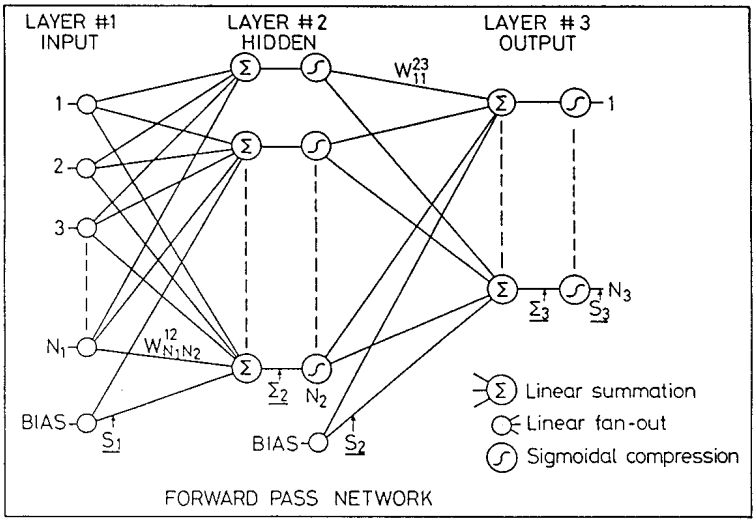


Figure 6: Schematic of the 3-layer neural network. $\underline{S}$ are the output vectors, $\Sigma$ are the input vectors for the different layers.

Why should such a structure interest us apart from its neurological analogy? Firstly, this neural network transfer function is extremely general. Kolmogorov[4] showed that any differentiable function can be reproduced by the superposition of simple "elemental" subfunctions. Hecht-Nielsen[5] showed that a 3-layer network is capable of forming this Kolmogorov type superposition to any arbitrary degree of accuracy, given enough hidden layer neurons. Extensions to more than one hidden layer are optional. Such a deeper network may require fewer weights to obtain a given accuracy, but it is not essential to make this extension in order to guarantee the existence of a mapping. The first advantage is therefore that the network transfer function is general and requires no design by the user other than the specification of the number of neurons in each layer.

Secondly, the calculation implied by this kind of multilayer neural network is parallel by nature and will be most suitable for massively parallel computation in the future, and is easily encoded in fast coprocessors at present. The algorithm will therefore be suitable for many powerful applications once implemented.

## IV.2 Learning

Having established the possible applicability of the neural network to our problem of estimating the $\sigma(D_r,T)$ mapping, we must find the weights $\mathbf{W}^{12}$ and $\mathbf{W}^{23}$ which best reproduce the given data set {Dr, T, σ}. The neural network transfer function is of course completely analytic, given by

$$\underline{S}_3 = \frac{1.0}{1.0 + \exp - \mathbf{W}^{23} \cdot \left[ \frac{1.0}{1.0 + \exp - \mathbf{W}^{12} \cdot [\underline{S}_1, 1.0]}, 1.0 \right]}$$

where $[\underline{x}, 1.0]$ implies the extension of the vector $\underline{x}$ by the scalar 1.0 bias term. We can define the optimal coefficients to be those such that the sum $Q = \Sigma_k \; | Dr_k - \underline{S}3_k |^n$ over the k available data examples be minimal. Alternatively we can minimise the relative errors, by renormalising. $Dr_k$ is the desired output corresponding to the inputs $\sigma_k$ and $T_k$, $S3_k$ is the calculated output. n=2 defines the commonly used least squares *L2* minimisation.Using these expressions, we must minimise the penalty function $Q(\mathbf{W}^{12}, \mathbf{W}^{23})$.

Two approaches for obtaining the optimal coefficients are possible. Firstly, for small matrices, we could present the function Q to a standard minimising library routine and obtain the two weight matrices. Alternatively, we can use the minimising techniques developed specially for neural networks.

As one aim of this study is to examine the applicability of neural networks, we have taken the latter choice, using the generalised delta rule of back propagation which is a form of gradient descent minimisation[3]. The weight matrices are initially set with random chosen weights in the range ±1.0. The set of examples is evaluated by the neural network and the output errors are averaged. All the weights are then updated according to established recipes,

and this procedure is reiterated until some convergence criterion is satisfied. A particularity of this method is its local behaviour, in that each network weight is modified by a quantity dependent only upon values available to the node it activates. Such a restriction is, in principle, irrelevant to our purposes, but will be an essential element in the realisation of truly parallel calculations which must minimise re-transmission of data within a chip, or between processors. The price to pay for this requirement is the inefficiency of the algorithm, compensated by its relative simplicity coded in a few lines of a high level language.

In order to apply the generalised delta rule to the neural network optimisation, we need to specify only a few learning parameters as well as a randomly distributed initial set of weights $\mathbf{W}^{12}$ and $\mathbf{W}^{23}$ and the size of the hidden layer. The learning parameters are the learning rate, set at 0.01 for $\mathbf{W}^{23}$ and 0.005 for $\mathbf{W}^{12}$ and the momentum term, set at 0.99, giving a roughly 100 full data set cycles "time constant" for the adaptation of the weights. The small values chosen for the learning rate are partly due to the fact that the relative error is always larger than the absolute error in the $[0,1]^n$ cube.

The output variable was taken to be Dr/10 to remain within the range of the output sigmoid, i.e between 0 and 1. An offset of 0.2 was added to the resultant value in order avoid an output close to 0, requiring large negative inputs to the output sigmoid. This in fact appeared to have little effect on the representation quality. The back propagation algorithm was adapted to minimise the relative error at the output considered to be more relevant to the experimental problem.

During these studies, various phases of the convergence procedure were found to be optimised by different sets of learning parameters, although the more efficient the less robust, in general. We do not discuss the learning optimisation in this paper.

**IV.3 Network size**

A first attempt was made to train a 5 neuron hidden layer network using a PC utility program[3]. Convergence was obtained quickly, but the residual errors were considered to be excessive.

Following this first attempt, a 15 hidden neuron network was trained using the same learning parameters. The results were far more encouraging. The relative errors of all 114 training examples is shown in the ($\sigma$:T) plane in Fig. 4b. The relative errors are also histogrammed in Fig. 5b, with an RMS value of 2.5%. Comparing with Figs. 4a,5a we see that the quality of the mapping is superior to that derived in the series expansion of Section III. The number of free parameters in the optimisation is 4 x hidden-neurons + 1 = 61 for this case.

To know how many hidden layer neurons are really necessary, we proceeded to reduce the size of the network. After a successful convergence, the active hidden layer neurons were suppressed in turn and the the RMS relative error recalculated for each case. The neuron whose suppression had the least effect on the RMS was then eliminated, and the new network allowed to re-converge starting with the relevant part of the previously obtained $\mathbf{W}^{12}$ and $\mathbf{W}^{23}$ weight matrices. This procedure was repeated until only 7 hidden layer neurons remained.

The degradation of the quality of representation was progressive. Fig. 7 shows various parameters determining the quality of the different networks : RMS relative errors, maximum relative error, maximum absolute error and the number of training examples reproduced with a relative error greater than 5 %.

The noise on this curve is due to the indecision concerning the final convergence. Most cases converge slowly towards the end of their learning, due to the disappearance of useful gradients along which the method descends. It is never clear when a final minimum has been approached, as can be seen in Fig. 8. We have plotted the convergence of the sum of the squares of the errors for the 10 hidden neurons and 7 hidden neuron cases. The 7 hidden neuron network learning seems to have converged, when it suddenly "feels" the

attraction of a previously "unsuspected" minimum, following which the sum of the squares drops by a factor of 5. This uncertainty is not always cured by an increase in gain, since this can cause the learning of other weights to oscillate. The only remedy seems to be to continue the training for a very long time.

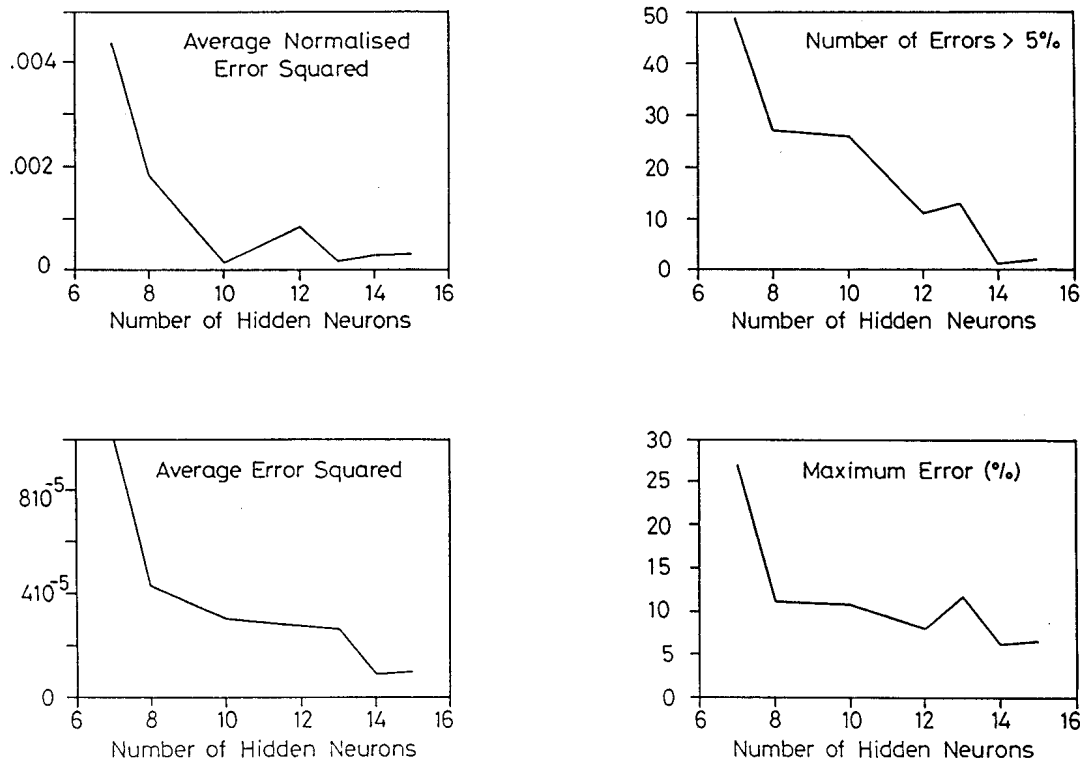


Figure 7: Degradation of the neural network mapping as the number of hidden layer neurons was reduced.

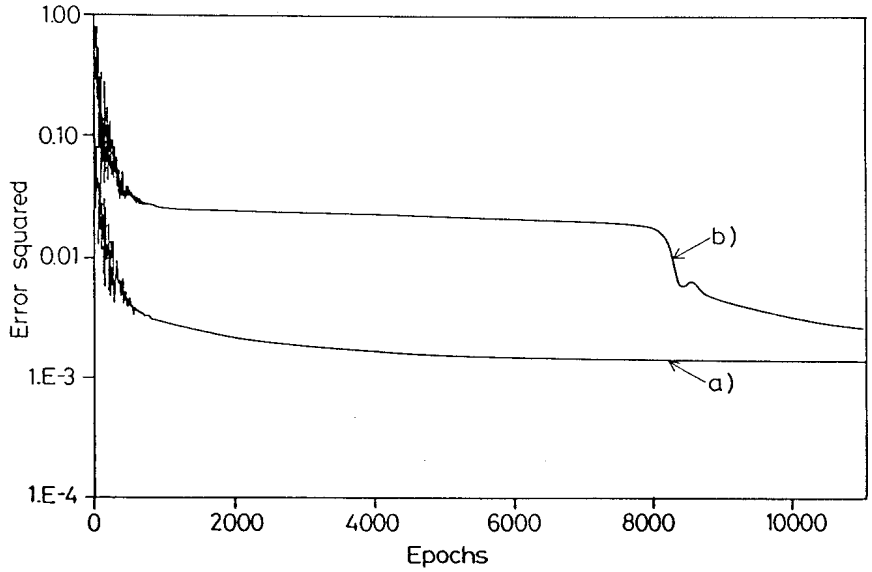


Figure 8: The convergence of the learning as a function of the number of full training-set learning cycles: a) 10 neurons and b) 7 neurons in the hidden layer.

When the number of hidden layer neurons is limited to 7, we find a systematic and smooth variation of the error surface. This itself implies that there is an inadequate number of free parameters. This behaviour is similar to that obtained if the series expansion were to have an inadequate order. Once the error surface is randomised, we can be sure we have approached the validity of the data points, and adding more hidden neurons can even falsify the neural interpolation, as in series expansion fitting.

Having seen the effect of varying the number of hidden layer neurons, we look at the effect of destroying separate individual weights, as an indication of the robustness of the network. We took the 15 hidden neuron solution and set each weight to zero in turn. The resulting RMS relative error is histogrammed in Fig. 9. The maximum RMS error, 1600 %, is a measurement of the robustness of the network. This result is disappointing, showing a clear distinction between weights which define the crude structure and weights which correct the fine structure. It is possible that the mapping information would distribute more evenly after prolonged convergence, but it seems that the back propagation algorithm is reluctant to allow other neurons to take over from a dominant neuron. It appears at first sight that a degree of robustness could be forced on the network by a modification of the learning algorithm.

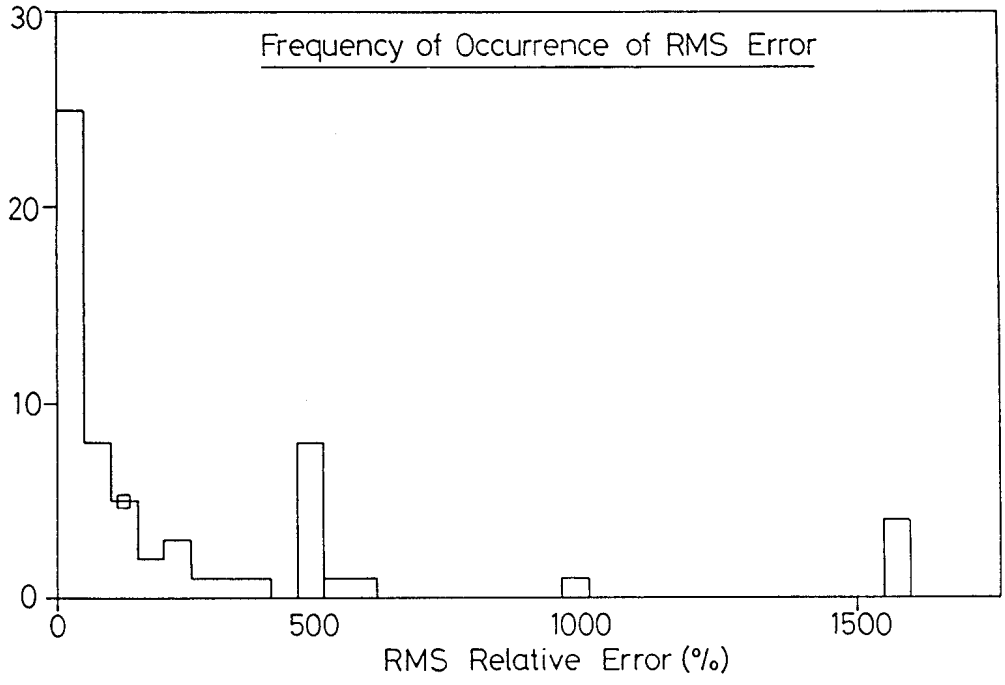


Figure 9: Histogram of the RMS relative errors, resulting from setting to zero in turn the weights of the 15 hidden neurons solution.

## V DISCUSSION

The classical method gave an adequate description of the functional relationship $Dr(T,\sigma)$, using 24 fitted parameters. In order to achieve this, the function had been pre-divided by the dominant "shape". An order of fit had to be chosen to describe each curve of Fig. 3 and an order of fit chosen to define the inter-curve dependence. These separate steps implied inspection by and knowledge of the "user". Such an inspection plus trial and error fitting requires a considerable investment of effort, of the order of weeks. This is why we asked the question: Couldn't all this be done more easily and faster with neural networks?

The neural network with 15 hidden neurons has 61 free parameters. Pre-dividing the data by the dominant bell shape allowed us to to fit well with fewer neurons, but this hybrid approach was rejected. The user only has to select the number of hidden neurons and the remainder was automated, provided the learning parameters were chosen to be conservative. The advantage of using a generic algorithm is considered to be substantial, especially in the future when specific parallel hardware becomes available. The use of these techniques appears promising for more complex problems, having used them in this relatively simple, but not specially chosen, illustrative problem. At no time in the fitting is use made of any knowledge of the function.

It has often been said that neural networks are robust in the sense that the failure of one neuron would not degrade the network function. This is not found to be true in our application. The neurons are not equivalent, some are of "low order" and are essential as are low order terms in a series expansion, others are of "high order" and their failure has little effect. The robustness of a network can usefully be defined as the largest relative error due to the failure of a single neuron weight, or of a single neuron.

Our purpose was to see whether the neural network can be applied in practice, rapidly and without previous knowledge of the characteristics of the function to be fitted. This "black-box" approach is only limited by the user having to choose the number of hidden neurons and to decide when convergence is attained. There are some trials necessary to fix the learning

parameters but the major time is consumed in computing. The user is not required to input any prior knowledge or skills.

Real time applications range from the linearisation of probe signals to the feedback control of complex processes needing many signal inputs and actuator outputs.

## ACKNOWLEDGEMENTS

It is a pleasure to acknowledge mainly helpful discussions with Robert W. Means on the use of artificial neuronal systems and with Prof. Froidevaux on developing the classical data analysis procedure.

Part of this work was supported by the Fonds National de la Recherche Scientifique.

## REFERENCES

1. G. Bugmann and U. vonStockar, Helvetica Physica Acta, **62**, 314-317 (1989).
   G. Bugmann and U. von Stockar, Trends in Bio-Technology, **7**, 166-169 (1989).
2. R.P. Lippmann, IEEE ASSP Magazine, April 1987, p. 4-22.
3. David E.Rumelhart and James L. McClelland , Parallel Distributed Processing: Explorations of the Microstructures of Cognition, Vols I,II &III, MIT Press (1986 & 1987).
4. A.N. Kolmogorov, Dokl. Akad. Nauk USSR, **114**, 953-956 (1957).
5. R. Hecht-Nielsen, Proc. 1988 INNS Annual Meeting, San Diego, USA.