Progress In Electromagnetics Research, PIER 18, 209–228, 1998

NON-LINEAR INTERFERENCE CANCELLATION TECHNIQUES FOR ELECTROMAGNETICALLY DENSE PROPAGATION ENVIRONMENTS

S. Ponnekanti and S. Sali

Department of Electrical and Electronics Engineering
University of Newcastle
Newcastle upon Tyne, UK NE1 7RU

1. INTRODUCTION

In certain signal processing applications such as radar and mobile communications, the desired signal is usually contaminated by both active and passive interferences. Examples of passive interferences include

radar clutter, atmospheric reverberation and specular noise. Active interferences are caused by the nonlinearities in the transmitter (such as class C-amplification in satellite transponders). Moreover, the presence of additive impurities at the sensors also add further non-Gaussian noise components at the receiver. Efficient forms of adaptive interference cancellation and channel identification in such hostile environments are, of significant interest to research community.

Traditionally, linear interference cancellation techniques are widely employed to accomplish the interference suppression. However, there are several applications where the linear filtering concepts are often too restrictive to yield satisfactory performance and the mathematical modelling has to be altered to accommodate non-linear filtering. Furthermore, in many interference cancellation systems in radar and digital mobile communications signal tracking and detection involves determining the signal from a sequence of several noise corrupted data which is subjected to non-Gaussian harmonics. Performance of linear filtering techniques in such correlated interference environments is poor and the situation demands the use of nonlinear estimation techniques. Generally the performance of any estimation scheme, depends on the observability of the information content of the data. This factor is governed by Signal to Noise and Interference Ratio (SNIR) and the relationship between the receiver and the target location. In other words, the estimation process is further encumbered by uncertainties associated with the poor observability.

Under certain scenarios, with suitable modifications in the modelling, Kalman Filters and Maximum Likelihood Method (MLM) [1–9] seem to perform well. However because of their massive parallelism and fast learning architectures Artificial Neural Networks (ANNs) offer greater potential [10–15].

Recently, ANN emerged as an alternative to the nonlinear estimation problems owing to the inherent simplicity and ability to approximate functions with non-linear behaviour. It is expected that an ANN can alleviate many of the difficulties existing with the current estimation methods. The generic neural networks and the node models are shown in Fig. 1 and Fig. 2 respectively. ANN is currently being explored in several applications including: Signal Processing, Control Pattern Recognition, Medicine Speech research, and Business. The advantages of the ANN for array signal processing applications can be outlined as follows: i- ANNs have the ability to learn from the lim-

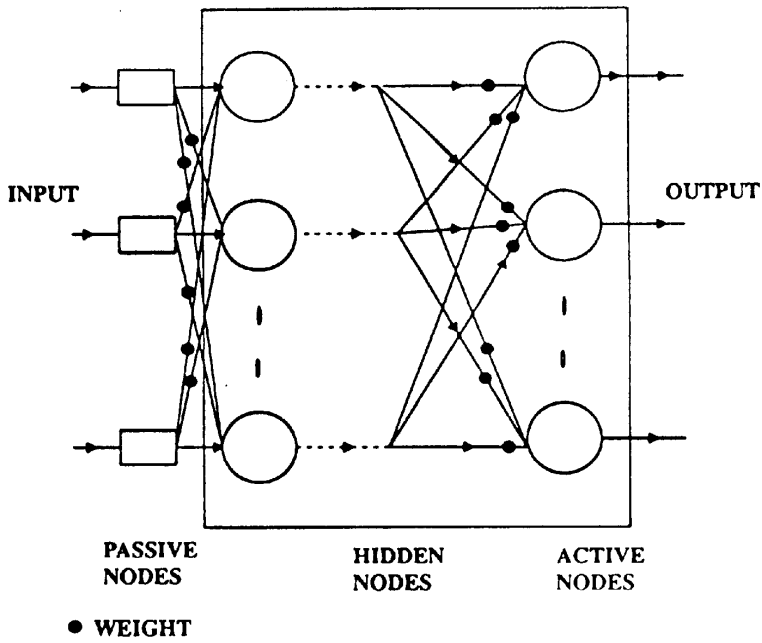

Figure 1. Multilayer Neural network.

ited samples of the input data, ii- ANN is a non-parametric technique and makes far less assumptions about the data distributions than traditional statistical methods, such as MLM, and finally iii- ANN has the capability to form highly non-linear decision boundaries for the portioning of the input space.

2. MODELLING OF NON-GAUSSIAN CHANNELS

K-distributed interference is gaining increasing importance for non-Gaussian and nonlinear propagation mediums and they are widely used in the statistical modelling of several communication and radar environments. The performance assessment of ANN for signals subjected to K-distributed interferences is critical from point of view of the non-linear interference cancellation. Because of this it is adopted in our studies as test bed to study the performance characteristics of the ANNs. With advent of new robust adaptive algorithms, there is an ongoing active research in this topic [3, 4].

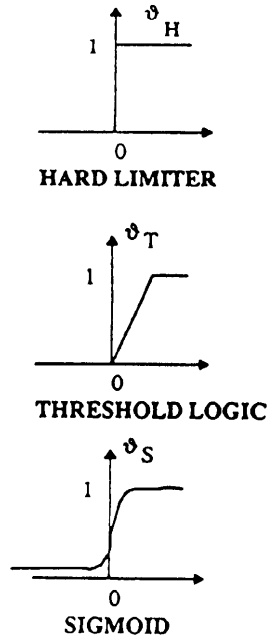

Figure 2. Node models for neural networks.

2.1 K-distributed Interference

Details of K-distribution statistical models are described in several articles [2–5]. K-distribution can be regarded as a compound statistical model . This stems from the fact that, the individual interference samples are Rayleigh distributed, while their mean is also a random variable following a Gamma Probability Density Function (PDF) [5]. The former is called a 'speckle' component and the later as a modulating component. This also amounts to treating the baseband equivalent of interference returns as a product of two mutually exclusive processes. The deviation from the usual Gaussian distribution is visualised, by considering signal under investigation, resulting from a superposition of a small number of equally important contributions. In this case central limit theorem cannot be applied, and hence Gaussianity is not guaranteed. K-distribution is claimed to be a fairly good fit to channels with the non-Gaussian interferences [6, 8, 9].

Theoretically, K-distribution is derived by averaging the speckle component over all possible values of the local mean level [3].

$$P(x) = \int_{\text{all } y} P(y)P(x/y)dy \tag{1}$$

where $P(x)$ is the over all probability density function (PDF) of the clutter returns, $P(y)$ is the PDF of the clutter mean level and $P(x/y)$ is the PDF of the speckle component. $P(y)$ is generally a good fit to the Chi family of distributions and is given by

$$P(y) = (2b/\Gamma(\nu))(by)^{2\nu-l} \exp[-b^2y^2] \qquad 0 < y < \infty \tag{2}$$

where b is a scale parameter, Γ denotes gamma function and ν is a shape parameter of the distribution. Also the speckle component, which has a Rayleigh amplitude distribution $P(x/y)$ of mean level y is given by

$$P(x/y) = (\pi x/2y^2) \exp(-\pi x^2/4y^2) \qquad 0 < x < \infty \tag{3}$$

The overall amplitude distribution is $P(x)$, of a K-distribution is given by

$$P(x) = (4c/\Gamma(\nu))(cx)^{\nu} K_{\nu-1}(2cx) \tag{4}$$

where $K_\nu(x)$ is a modified Bessel function, c is also a scale parameter related to the distribution. Further, the n^{th} moment is given as

$$\overline{x}_n = (1/c^n)(\Gamma(\nu + n/2)/\Gamma(\nu))\Gamma(n/2 + 1) \tag{5}$$

The cumulative probability is also given by

$$P_c(x) = (2c^{\nu})/(\Gamma(\nu))t^{\nu} K_{\nu}(2ct) \tag{6}$$

The value of the inverse shape parameter $(1/\nu)$ is bound by an upper limit of 0.4 and a lower limit of 0.3.

2.2 An Efficient Method for the Generation of K-distributed Variables

The n^{th} moment of K-distribution is taken from equation (5) and from that first and second moments are obtained as mean and variance as follows:

$$m_x = (1/c)[\Gamma(\nu + 1/2)/\Gamma(\nu)]\Gamma(3/2) \tag{7}$$

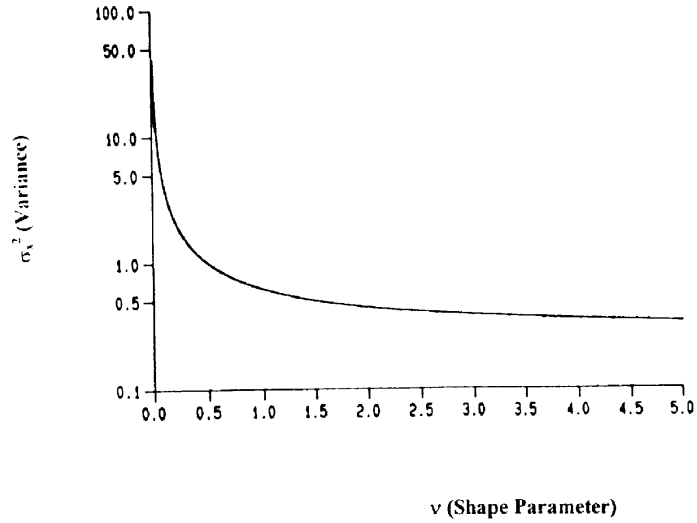

Figure 3. Relationship between the variance and shape parameter.

The second moment can be written as:

$$\sigma_x^2 = E[x^2] - m_x^2 \tag{8}$$

Correspondingly, the second moment of K-distribution is given by

$$\sigma_x^2 = (1/C^2)[\Gamma(\nu+1)/\Gamma(\nu)]\Gamma(2) - m_x^2 \tag{9}$$

Assuming unit mean process (9) may be rearranged to yield

$$\sigma_x^2 = [\Gamma(\nu+1)\Gamma(2)\Gamma(\nu)\Gamma^2(\nu+1/2)\Gamma^2(3/2)] \tag{10}$$

A graph illustrating the relationship between the variance and the shape parameter is plotted in Fig. 3. We can see that, the variance is decreasing for increasing shape parameter value. At this stage, we can choose any value of variance and obtain ν from the graph and obtain the corresponding c from (7). We also have from the cumulative distribution function in (6) repeated here for convenience.

$$P_x(x) = 1 - ((2c^\nu)/(\Gamma(\nu)))\ x^\nu\ K_v(2cx) \tag{11}$$

At this stage, a table of x vs $P_x(x)$ for all possible values of x_i is created. To generate the K-distributed variables from this table search algorithm takes a random number from standard uniform random number generator and by interpolation and search, it finds out the corresponding $P_x(x)$. The accuracy of this random number generator depends upon the sample size and usually large sample size is recommended to maintain the distribution accuracy. After generating the K-distributed variables. the ensemble runs are performed to check the mean and variance of the samples. It is found that, with a large sample size, the theoretical mean and variance are approximately equal to the mean and variance of the simulated random variables. For example ensemble mean from simulation is equal to 0.994879 and given theoretical is equal to 1.0 and ensemble variance from simulation is 0.386324 and the theoretical theoretical value is 0.3833. It can be noticed that, both the simulation values and the specified values are in good agreement. Finally, it should also be noted that, this is by no means the only method and there are various other ways of generating the random variables. Details of various methods can be consulted in the literature [5, 8].

3. APPLICATION OF NEURAL NETWORKS

The application of neural network for channel identification and signal detection involves classification and training stages. The goal of a neural network classifier is to train the net to achieve a balance between the ability to respond correctly to the input signal that is used for training and the ability to offer good response to the signals during the testing phase. The former is called *memorisation* and the latter is called *generalisation*. The role of the training sequence is to adapt the network to fit into the desired response using a stochastic gradient test procedure. It should be noted that, during the testing phase, the signals presented to the net need not be identical to the signals appeared during the training phase. There are several ways of designing an ANN classifier, namely, Back propagation or Generalised delta rule based classifier [10], Radial Basis Function (RBF) based classifier [16], Generalised Probabilistic Neural Networks (GPNN) [17] and Cascade Correlation Neural Networks [12].

3.1 Radial Basis Function (RBF) Networks

A multilayer neural network representation adopted in these studies is shown in Fig. 2, which shows the node model of the ANN. Sigmoid function is usually assumed in the literature for the node model. Among the nonsigmoid activation functions, RBF is the most frequently used function [3–5]. The hidden layer projects the input space x into another space ξ. This projection is also strictly bound to $(\xi_l, \xi_2, ..., \xi_n) \in \xi$, where n is the number of hidden nodes in that layer. The mapping performed by each hidden node is described by $[-1, 1]^n$ and is expressed as

$$\xi_k = \exp\left[-\sum_{i=1}^{l} \frac{(x_i - m_{ki})^2}{2\sigma_k^2}\right] \tag{12}$$

where l is the number of components in each input vector, $(x_1, x_2, \ldots, x_l) \in X$ and ξ_k denotes the output of the k^{th} hidden layer. From linear space theory, σ_k's are the coefficients used in computing the weighted norm, given by

$$\xi_k = e^{\langle x - m_k \rangle_{\sigma_k}^2} \tag{13}$$

where $\langle\ \rangle$ denotes a weighted Euclidean norm This weight norm denotes the radial distance from any vector x, to the centre m_k. Further, each output node finally computes the weighted summation of the values passed on by the nodes in the hidden layer, given by:

$$y_j = \sum_{i=1}^{n} w_{ij}\xi_i \tag{14}$$

where W_{ij} are the weights. It may be noted that the term basis in the RBF indicates that, the output lies in the linear space spanned by the outputs of the hidden layer nodes.

3.2 Kohenen Networks

In some array processing applications the desired signal may not be available or may be lost in high interference environments. In such scenarios the receiver systems have to be trained blindly in the absence of the desired signal. Recently, Kohonen Self Orthogonalising Networks (KSON) emerged to cater the needs of applications where the

training sequence is not available. The Kohonen Self Orthogonalising Method (KSOM) [15] is an algorithm for detecting features in the received data and partitioning the data space according to the features. The networks of Kohonen attempt to mirror the orderly placement of neurons in the sensory pathways of the brain in their representation of the external stimulus. No information is used initially about the correct classification of the input data patterns; instead, the KSOM discovers its own natural classification scheme based on the combined distribution of the patterns within the high dimensional data space.

In the basic structure of the KSOM, all input units are connected to an output grid. The output units are extensively interconnected with many local connections. The result of this lateral interaction is that, after adequate self-learning steps, the network tends to be spatially organised according to the structure of the input data set. The units become tuned to specific input vectors, or groups of them, such that each unit responds only to some specific patterns in the input set. In addition, the weights are organised such that topologically close nodes are sensitive to inputs that are physically similar.

3.3 Data Processing in ANN

The performance of the ANN, trained with the standard Back Propagation algorithm, is heavily dependent on the input data representation. In this section, we outline few data processing methods. used in the context of application of ANN, which is also called pre-processing. In array signal processing, input vector mainly contains raw data from sensors. The data samples consist of inphase and quadrature signal components, the amplitudes of the interference signals, and clutter. It is observed that, application of these inputs directly to the network yield unacceptable performance. This is due to the fact that the raw data conveys only first-order information with a certain degree of inherent randomness. This has to be minimised to ensure acceptable performance of the classifier, and there is, therefore, a need for appropriate pre-processing. This is also called the feature-extraction stage. However, finding a suitable representative feature, in itself, is a difficult problem. A pre-processing method is developed in [9]. This procedure determines the phase difference between received data by consecutive sensors and then utilises the sine and cosine of this phase difference as the inputs to the ANN. Alternatively, a ML method can be used in the pre-processing stage to supply the time-delays or angles to the ANN [1,

17]. The ANN can be utilised to learn from these transformed inputs. Because of the computational complexity involved in these schemes, we adopt a more flexible alternative which employs spectral analysis concept based on Discrete Fourier Transform (DFT). In this method, the normalised spectral coefficients are derived via DFT and coupled with a suitable window function for use as features on which an ANN classification is based [11].

3.4 Training Algorithm

One of the most widely employed ANN is based on the most popular Back Propagation [13]. This training scheme directly evolved from the iterative gradient search algorithms. The hub of these algorithms is to minimise the sum of the squared error ξ, which is the difference between the desired output (D-vector) and the actual output (Y-vector) of the Multilayer Neural Network. This error is given by

$$\xi = \frac{1}{2} \sum (D - Y)^2 \tag{15}$$

In the back propagation algorithm, as in the case of other adaptive algorithms, the error minimisation is done through selection of suitable weights. The crucial requirement is the node transfer function. The designer has to select, only, the function that is differential everywhere [12]. Any one of the activation functions, shown in the Fig. 2 can be selected. Usually, sigmoid function is utilised as an activation function in the literature [12]. Training a Neural Network by back propagation involves 4 distinct stage: i- the input patterns from the selected problem, ii- the feeding forward of the input pattern through the network, iii- the back propagation of the associated error and iv- updating the weights in the network. During the feed forward, the input signal received by each input node is passed on to the hidden layer nodes. The response of the hidden node is then computed and transferred to the output node in the output layer. Each output node, in turn, evaluates its response and compares it with desired output value supplied by the designer. The resulting error is sent back to the nodes in the hidden layer that is connected to the output. The weights between this hidden layer and the output are also adapted based on this error, at a later stage. In a similar manner, all the hidden layers receive the back propagated error. After all nodes in the layers are completed. the weights of all the layers are adjusted simultaneously [2, 14]. Gradient

algorithm used in the weight adjustment is usually Least Mean Square (LMS) adaptive algorithm [2] and it also formed the basis of training algorithm in studies presented in this paper.

4. PERFORMANCE EVALUATION OF ANN WITH K-DISTRIBUTED INPUTS

In order to evaluate the performance of the ANN, we conducted several computer simulations. K-distributed noise samples are used at the input of the ANN. Several initial simulations are conducted to evaluate the most optimum values for the LMS convergence and momentum factor. Values of 0.7 for the former and 0.1 for the latter yielded the best performances and used in our studies throughout. The optimisation of these parameters is not attempted. In all the cases, the network architecture has the same format as illustrated in Fig. 1, i.e., the network is specified by **Input units - hidden layer(s) units - output units**. For example, all the case studies for processing the K-distributed noise, employed a 40-30-2 architecture. As a result, this architecture is modelled with 40 input units, one hidden layer (with 30 units) and two output units. Therefore, two output units are used with the teacher being (0.9,0.1) and (0.1,0.9) for representing signal and noise respectively. A single hidden layer is used. Although much research has been carried out in determining the optimum number of hidden units, the results are all applications dependent. Therefore, networks with varying numbers of hidden units (from 5 to 30 in steps of 5) are trained in an effort to find the near-optimum architecture suited to the problem at hand.

Each network is simulated until the mean square error on the training set is minimum. This point is reached in approximately 25 training epochs. The performances, expressed as a percentage, on the training and the test sets for the six different networks are shown in Table 1. Very high classification rates are exhibited by all networks during both training and testing, the best testing performance being 96.01% by network 40-15-2. However, the influence of the number of hidden units, in this architecture, is not overwhelming. This conclusion is also supported, by the error convergence studies. Typical error convergence curve for the best architecture is shown in Fig. 4 where, after few initial epochs, the convergence is reached very quickly. Fig. 5 also depicts the learning performance of the best network compared with a 40-10-2 network. During the discrimination between pure signal and

Network	Training	Testing
40-5-2	97.67	95.85
40-10-2	100.00	95.68
40-15-2	98.33	96.01
40-20-2	98.33	95.85
40-25-2	98.33	95.51
40-30-2	98.67	95.85

Table 1. Performance of different Multi Layer Perceptrons (MLP).

noise, the percentage of correctly classified signal patterns for various values of noise power (P) is shown in Table 2 for three classification criterion. A criterion of 0.03 means that a pattern is considered as being correctly classified only if the least mean square error between the actual network output on presentation of this pattern and the desired output is less than 0.03. Thus, the lower this value, the more stringent the criterion. The 'maximum' criterion on its part only yields correct classification if the indices of the maximum values in the actual and the desired output vectors on presentation of a pattern are the same. The choice of a particular criterion for quoting performance figures may be a subject of discussion, but it is believed that such criterion should be relatively less stringent as the problem complexity increases, i.e., as the noise power relative to the signal increases. The robustness of the MLP is clearly demonstrated by the 'maximum criterion' curve of Table 2 in which it achieves nearly 50% recognition even at a noise power of 0.9. A series of computer simulations revealed that, the number of hidden layers plays a rather dominating role during the initial periods of convergence.

5. PERFORMANCE EVALUATION OF ANN WITH REAL-TIME CLUTTER INPUTS

In this section, we outline the performance of ANN with real-life clutter interference as inputs to ANNs. Radar clutter is selected as an example to represent a real-life non-Gaussian interference in electro-

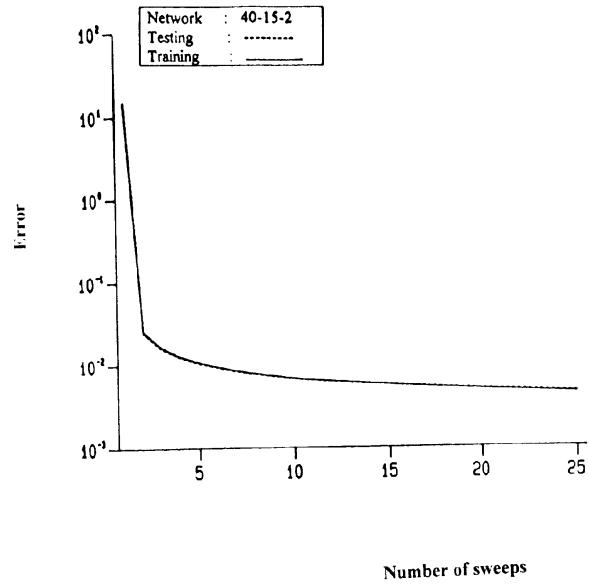

Figure 4. Error convergence of neural networks.

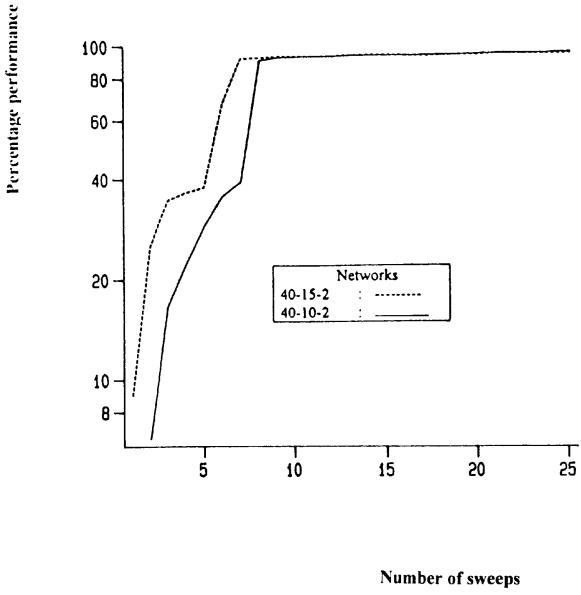

Figure 5. Learning performance of neural networks.

Noise Power	Criteria:0.03 Performance%	0.06 Performance%	Maxim Performance%	Kohonen Performance%
0.01	100	100	100	100
0.1	96.01	100	100	97.78
0.2	42.35	42.35	100	65.41
0.5	0	0	84.26	60.23
0.7	0	0	69.84	58.45
0.9	0	0	47.67	55.80
1.0	0	0	38.58	55.43
1.2	0	0	30.82	35.45
1.5	0	0	21.29	25.06
1.7	0	0	15.96	10.55
1.9	0	0	12.64	6.45
2.0	0	0	10.42	5.76

Table 2. Performance of the network using 3 criterion.

magnetically dense environments. We reconstructed some measured clutter data including some nonlinear harmonics from the transmit system which used class-C amplification [6], resulting in a very small sample size of 100. The training and testing ANNs on these limited data set is proved to be a formidable task. Because of this, we adopted a pre-processing approach which consisted of segmenting the data into small windows. This segmentation is done carefully to ensure that, we end up with some signal-only and noise-only windows. This process, further reduced the data sample size, available for the ANN to receive training and testing. To overcome this problem, we resorted to overlapped windows. These overlapped windows are utilised to evaluate the performance of ANN. The error convergence and percentage performance is depicted in Table 3 during both training and testing. In all the cases, both the number of input nodes and the number of output nodes are kept fixed. The number of hidden units and hidden layers are both varied. Comparing the sets of 15-10-2 and 15-20-2 confirms the conclusion that the number of hidden units, given the fixed architecture, does not influence the error convergence significantly. Studies are also carried out to obtain the information about the influence of

Network	Training Error and Percentage Performance	Testing Error and Percentage Performance
15-10-2	0.0065,100	0.3022,50
15-20-2	0.0053,100	0.3038,50
15-10-20-2	0.0229,100	0.2039,50
15-20-10-2 (Low-Momentum)	0.0025,100	0.1595,70
15-20-10-2 (Med-Momentum)	0.0051,100	0.1070,80
15-20-10-2 (High-Momentum)	0.0035,100	0.1821,70
15-20-25-10-2	0.00458,100	0.1393,80
15-30-10-2	0.0049,100	0.1562,70

Table 3. Performance of MLP with real-life interference.

number of hidden units on the convergence of ANN. By comparing ANNs 15-10-20-2, 15-20-10-2 and 15-30-10-2, it is noticed that network with 20 hidden units in the first hidden layer performed well both in terms of error reduction as well as the percentage performance rather than its counter part with 30 hidden units in the first hidden layer. Finally, adding more hidden units to the first hidden layer did not improve the performance of the ANN. A specific example of error convergence during the testing phase for networks 15-30-10-2 and 15-20-25-10-2 is shown in Fig. 6, where the network with 20 hidden nodes in the first hidden layer clearly outperformed that of 30 hidden units. The percentage performance of both the networks during testing phase, depicted in Fig. 7, also confirms our conclusion. Several simulations, demonstrated the fact that the performance of ANN critically depends on the architectural specifications.

6. PERFORMANCE OF KOHONEN NETWORKS

Learning steps in KSOM is a two dimensional map of units, each of which carries a tag or label of the object that has excited it at the

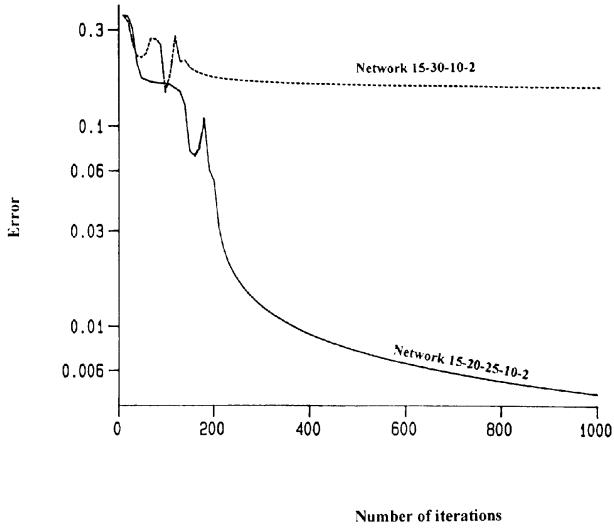

Figure 6. Error convergence during testing phase.

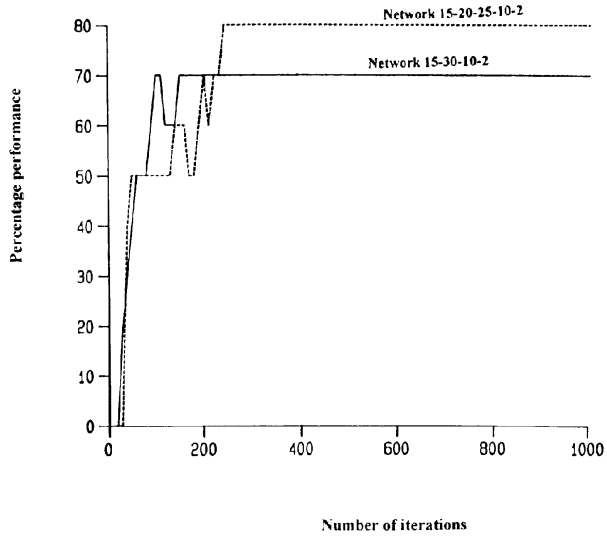

Figure 7. Comparative study of the percentage performance during testing phase.

Network	Training(%)	Testing(%)
5*5	74.00	75.08
8*8	86.67	87.38
10*10	94.67	95.35
12*12	90.33	94.19
15*15	88.67	89.70
18*18	90.00	90.37

Table 4. Performance of different Kohonen Networks.

final training epoch. Units excited by patterns of the same class form clusters on such a map. If a test pattern falls into a cluster, it is classified as belonging to the group which forms this cluster. An important consideration in the use of the KSOM is the choice of the size of the output layer. An excessively large layer produces empty regions with units not excited by any pattern and too small a layer causes conflicting situations where units are excited by patterns from different classes.

In order to determine the near-optimum layer size, a set of networks with different numbers of output units are simulated, these having dimensions 5*5, 8*8, 10*10, 12*12, 15*15 and 18*18. The results are shown in Table 4. These performance figures are only reached after about 200 training epochs which is in contrast to the 25 epochs for the MLP. This is simply explained by the fact that, contrary to the MLP, the KSOM has no teacher during the learning phase and thus requires more training epochs to extract the clusters from the input data. The network with the highest performance and producing the most distinct map, i.e., completely separate clusters with no empty regions, is the one with a 10*10 output layer size. This network is subsequently tested on data sets with increasing noise levels and the results are already depicted in Table 2. Sharp reductions in performance are observed for noise power 0.2 and 1.2. One important aspect of KSOM, is that unlike the MLP, it correctly classifies more than 50% of the signal patterns, corrupted with equal noise power (i.e., $P = 1$). It must be emphasised that KSOM performs in blind scenarios where MLP

will not produce any result and as far as robustness is concerned, the KSOM outperforms the MLP. As a general conclusion, it seems that unsupervised learning procedures are better suited to the extraction of signals and they cater for all the environmental scenarios. The longer training times encountered do not pose any major drawback since training can be carried out off-line and, once completed, the network is capable of almost instantaneous results, crucial in real time applications.

7. CONCLUSIONS

In this paper, we have dealt with the concept of non-linear interference cancellation in electromagnetically dense environments where the desired signal is contaminated by several unwanted non-Gaussian interferers. A new nonlinear interference canceller is introduced based on Artificial Neural Networks and the performance evaluations highlighted the importance of the network architecture on the performance. A simple and effective method of simulating K-distributed interference, which is known to provide a better model to the practical non-Gaussian electromagnetic interferences is devised. The performance of ANNs against the K-distributed noise inputs is addressed. Further, some examples of the noise cancellation capabilities of the back propagation based ANN are also illustrated. It was concluded that, selection of the architectural parameters is most crucial, to realise suitable receivers tailored to the practical applications. It is also demonstrated that, blind interference cancellation in the absence of desired signal can be accomplished by a class of Kohonen Self Orthogonalising Neural Networks.

ACKNOWLEDGMENT

The authors express their appreciation for the financial support given by the UK Engineering, Physical Science Research Council. The advise and software support on Neural Network provided by Dr. Y. Bisseur were greatly appreciated.

REFERENCES

1. Ponnekanti, S., and S. Sali, "Efficient antenna beaming," *Electronics Letters,* Vol. 12, March 1996.
2. Widrow, B.,"Neural nets for adaptive filtering and adaptive pattern recognition," *Computer,* 25–39, March 1988.
3. Watts, S.,"Radar detection prediction in K-distributed sea clutter and thermal Noise," *IEEE Trans.*, Vol. 23 AES, No. 1, 40–45, 1987.
4. Haykin, S. S., and W. Stehwien, "Classification of radar clutter in air traffic control environment," *IEEE Proc.*, Vol. 779, No. 6, June 1991.
5. Farina, A., A. Russo, and F. Scannapieco,"Radar detection in coherent weibull clutter," *IEEE Trans. ASSP,* Vol. 35, No. 6. 893–895, June 1987.
6. Gibson, C., and S. Haykin,"Radar performance studies of adaptIve lattice clutter suppression filters," *IEEE Proc.*, Part-F, Vol. 130, No. 5, 357–367, August 1983 .
7. Ponnekanti, S., and S. Sali, "Adaptive antenna techniques for mobile communications," *IEEE Proc. Int. Conf. on Wireless Comms.*, (IPWC 96), 243, 19–21, Feb. 1996.
8. Cramer, H., *Mathematical Methods of Statistics,* Englwood Cliffs, NJ, 1946.
9. Nutall, A. H., and G. C. Carter, "A generalised framwork for power spectral estimation," *IEEE Trans.*, Vol. 128, 334–335, 1980.
10. Simmers, J. A., and T. O'Donnekk, "Adaptive radial basis function neural beamforming," *IEEE Proc.*, (c31), ITA Conference, l09–118, June 1992.
11. Leonard, J. A., M. A. Kramer, and L. H. Ungar,"Using radial basis functions to approximate a function and its error bounds," *IEEE Trans.*, Vol. 3, No. 4, 624–627, 1992.
12. Fahlaman, S. E., and C. Liebiere, "The Cascaded-Correlation Learning Architecture," In D.S Touretsky, ed., *Advances in Neural Information Processing Systems,* Vol. 2, 524–532, 1990.
13. Nielseo, H., *Neuro Computing,* Addison-Wesley, 1989.
14. Rumelhart, D. E., and J. L. McClelland, *Parallel Distrubuted Processing (PDP): Explorations in the Microstructure of Cognition,* Foundations, Vol. 1, MIT Press, Cambridge, MA., 1986.
15. Kohonell, T., *Self-organization and Associate Memory,* Springer-Verlag, Berlin, 1984.
16. Park, J., and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Computation,* Vol. 3, 246–257, 1989.

17. Specht, D. F., "Probabilistic neural networks," *Neural Networks,* Vol. 3, No. 1, 109–118, 1990.
18. Parzen, E., "On estimation of a PDF and mode," *Ann Math. and Stat.*, Vol. 33, 1962.
19. Lipmann, R. P., "An introduction to computing with neural net," *lEEE ASSP MaGazine,* Vol. 4, 4–22, 1987.
20. McClelland, J. L., and D. E. Rulllelllart, *Explorations in Parallel Distributed Processing,* MlT Press. Cambridge. MA, 1988.