SRC TR 89-33

Cascaded Neural-Analog Networks for Real Time Decomposition of Superposed Radar Signals in the Presence of Noise

By

A. Teolis, Y.C. Pati,
M.C. Peckerar and S. Shamma

Cascaded Neural-Analog Networks for Real Time Decomposition of Superposed Radar Signals in the Presence of Noise

A. Teolis Y. C. Pati* M.C. Peckerar S. Shamma

Systems Research Center
University of Maryland
College Park, MD., 20742

U.S. Naval Research Laboratory
Code 6804
Washington D.C., 20375

Abstract

Among the numerous problems which arise in the context of radar signal processing is the problem of extraction of information from a noise corrupted signal. In this application the signal is assumed to be the superposition of outputs from multiple radar emitters. Associated with the output of each emitter is a unique set of parameters which are in general unknown. Significant parameters associated with each emitter are (i) the pulse repetition frequencies, (ii) the pulse durations (widths) associated with pulse trains and (iii) the pulse amplitudes. A superposition of the outputs of multiple emitters together with additive noise is observed at the receiver. In this study we consider the problem of decomposing such a noise corrupted linear combination of emitter outputs into an underlying set of basis signals while also identifying the parameters associated with each of the emitters involved. Foremost among our objectives is to design a system capable of performing this decomposition/classification in a demanding real-time environment.

We present here a system composed of three cascaded neural-analog networks which, in simulation, has demonstrated an ability to nominally perform the task of decomposition and classification of superposed radar signals under extremely high noise conditions.

1 Introduction

Addressed here is the problem of decomposing a linear combination (or superposition) of basis signals into its underlying components under the constraint that the superposition has been noise corrupted. We emphasize the fact that the basis functions need not be known apriori. Clearly a problem of this nature has direct applications in practical radar systems.

For instance, consider the scenario in which the radar environment is crowded with signals from a multitude of radar emitters. Each emitter propagates its unique parametric

*supported in part by NSF under grant OIR-85-00108

representation of its characteristic signal through the environment independently of the others. It is desirable to posseses the capability of determining both the number and identity of the emitters present. This type of identification or *classification* is probably most relevant in military applications where it is imperative to differentiate between friendly and hostile radars.

For a system designed for such an application to be practical, it is clearly necessary that intensive processing be performed. Hence, foremost among our objectives is the design of a system capable of performing this decomposition in a demanding real-time environment. In this respect, conventional digital hardware implementations are not likely to succeed. For this reason, we focus attention on an analog neural network solution instead. By exploiting the parallel nature of the neural network architecture, it is possible to far exceed the speed of an equivalent digital implementation. Moreover, the neural topology eliminates the need for an explicit algorithmic development.

Among the specifications for such a system are good estimation of the weights of the underlying basis functions when the input signal has been severly degraded by noise. It will be shown that the proposed system employs a non linear neural network noise reduction network which performs demonstrably better than a simple linear low pass filter. Non-linear noise reduction aides the classification process aiding satisfaction of the aforementioned specifications.

2 Problem Statement

Figure 1 indicates the model for the Radar Decomposition System. For analytical convenience this model is transformed by reflecting the sampling process all the way back to the introduction of the basis functions (see Figure 2). Such a transformation has the advantage of allowing the problem formulation to be carried out solely in *discrete* time. Also, a realistic signal processing application dictates that only a window (a finite number of samples) of a signal may be processed at any point in time. The number of points in this window is assumed to be fixed and is denoted as n. Thus, the domain of the signal can be defined as $\mathbb{N}_s \triangleq \{ 0, 1, \ldots, n-1 \}$.

Suppose there exits a finite set of linearly independent *discrete time* basis functions,

$$\mathcal{B} = \{ \phi_{\{0,\Theta_0\}}, \phi_{\{1,\Theta_1\}}, \cdots, \phi_{\{N-1,\Theta_{N-1}\}} \}. \tag{2.1}$$

Here $\phi_{\{k,\Theta_k\}} : \mathbb{N}_s \mapsto \mathbb{R}$ is a member of the basis set and $\Theta_k \in \mathbb{R}^M, k = 0, 1, \ldots, N-1$ is an M dimensional parameter vector indexing ϕ. Unless otherwise indicated, N will for the remainder of this paper refer to the number of basis functions in the basis set $\mathcal{B}$.

A signal[1] s, is generated as a linear combination of the basis set, where the corresponding weight vector, $\mathbf{a} = (a_0, a_1, \cdots, a_{N-1})^T \in \mathbb{R}^N$. That is,

$$s : \mathbb{N}_s \mapsto \mathbb{R} \ni s = \sum_{k=0}^{N-1} a_k \phi_{\{k,\theta_k\}} = \mathbf{a}^T \Phi(\mathcal{B}) \tag{2.2}$$

[1]Note that the signal s is actually the sampled signal obtained from the superposition of the *continuous* basis functions, however, we have assumed that basis signals themselves have already been sampled.

OR

$$s(t) = \sum_{k=0}^{N-1} a_k \phi_{\{k,\theta_k\}}(t), \quad t \in \mathbb{N}_s \tag{2.3}$$

$\Phi(\mathcal{B})$ is the N-dimensional vector formed by the basis functions given as

$$\Phi(\mathcal{B}) = (\phi_{\{0,\Theta_0\}}, \phi_{\{1,\Theta_1\}}, \cdots, \phi_{\{N-1,\Theta_{N-1}\}})^T. \tag{2.4}$$

Note that the linear independence of the basis set, $\mathcal{B}$, insures that the signal, s, has an unambiguous (unique) representation with respect to the basis set. Formally, a finite set $\mathcal{B}$ consisting of elements $\phi_k, k = 0, 1, \ldots, N-1$, has linearly independent elements if the following condition is satisfied:

$$\sum_{k=0}^{N-1} a_k \phi_k = 0 \quad \Leftrightarrow \quad a_k = 0, k = 0, 1, \ldots, N-1 \tag{2.5}$$

OR

$$\mathbf{a}^T \Phi(\mathcal{B}) = 0 \quad \Leftrightarrow \quad \mathbf{a} = \vec{0} \tag{2.6}$$

A second signal, $\tilde{s}$, is produced by simply adding Gaussian noise of mean μ and variance σ^2. Thus $\tilde{s}$ is the signal that is presented to the decomposition network, where

$$\tilde{s}(t) = s(t) + n(t), \qquad n(t) \sim \mathcal{N}(\mu, \sigma^2), \quad t \in \mathbb{N}_s \tag{2.7}$$

The process by which the input signal $\tilde{s}$ comes about is represented pictorially in Figure 1. We are now in a position to clearly state the problem which we would like to solve.

Problem (P): *Given the gaussian noise corrupted signal $\tilde{s}$, along with a linearly independent set, $\mathcal{B}$, of basis functions for s, recover the vector, $\mathbf{a}$, of weights.*

3 Overview of the System

To solve the problem (P) above, we propose a cascaded three block architecture. As depicted in Figure 2, the data flows from left to right and is processed serially by each of the three main components of the system. The first block consists of a MEDN to perform the task of noise reduction. Spectrum analysis of the output of the first block is performed in the next block by a second analog neural network. Finally, decomposition of the aggregate signal is performed in the third block by yet another analog neural network. This final classification is based on the output spectrum of the previous block.

Problem (P) comes about mainly from the physical realities of the radar application motivating this paper. As such, it would be instructive to qualitatively discuss and justify the nature of the proposed system. Presented to the system is the noise corrupted superposition of radar pulses as displayed in Figure 2. The objective of the system is to decompose this noise corrupted superposition into its underlying components or basis functions. Each basis function may be associated with a particular emitter. Since the

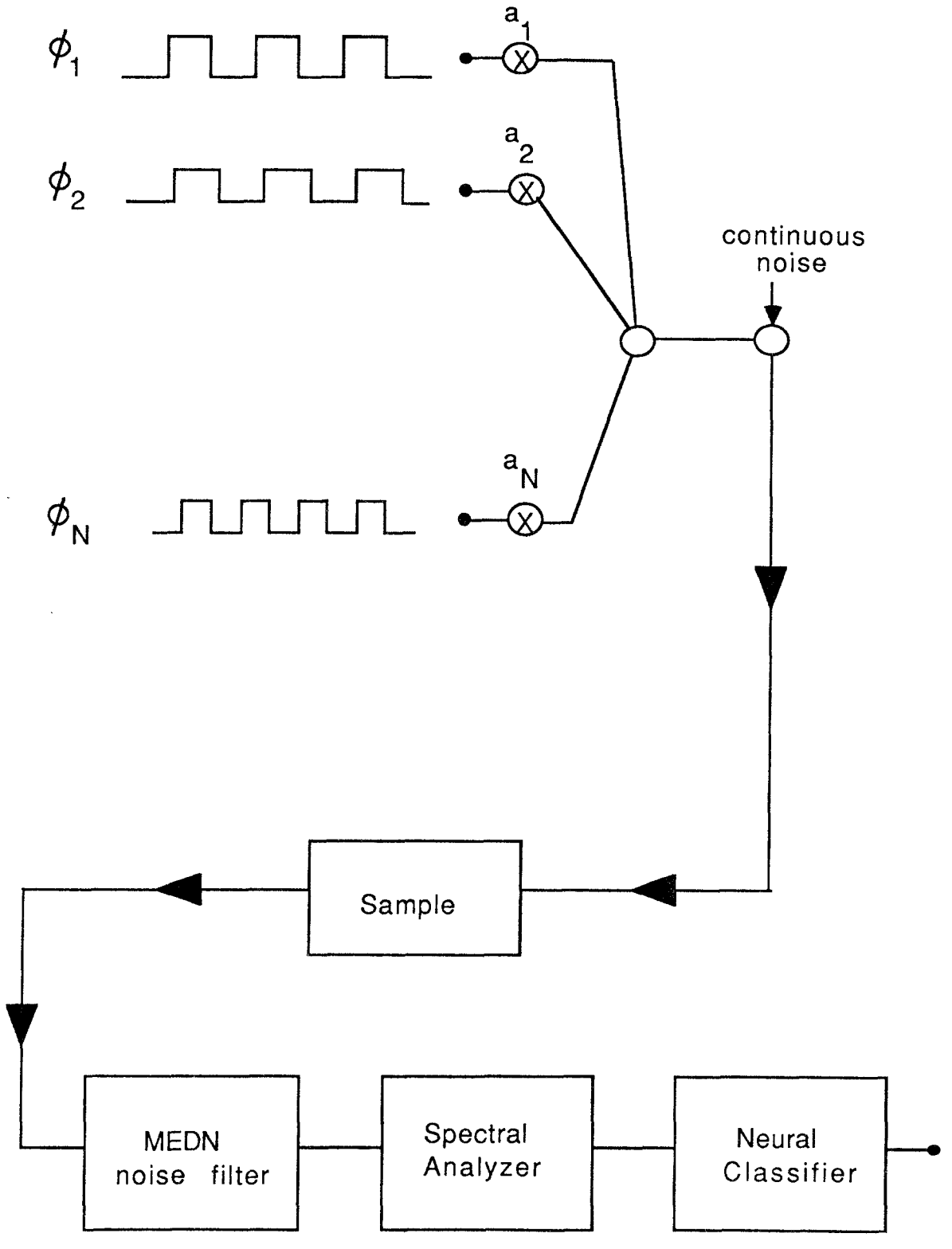


Figure 1: *Schematic of the Radar Decomposition System*

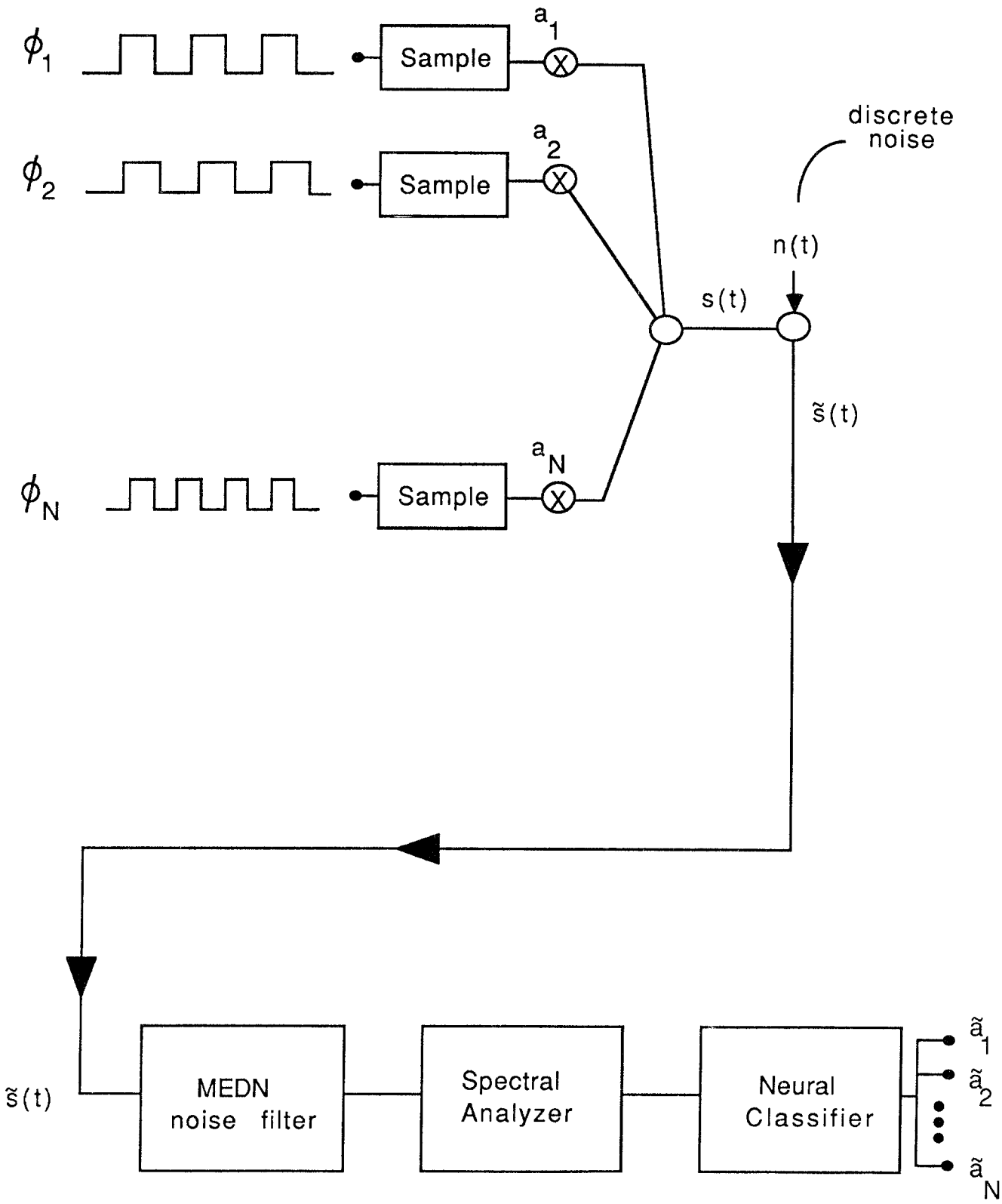


Figure 2: *Schematic of the Radar Decomposition System where the sampling is reflected to the basis functions*

presented signal is corrupted by noise, it is appropriate to attempt to reduce that noise level in the hope that more reliable data will facilitate the ultimate goal of decomposition. Hence, the MEDN noise reducer preprocessing is justified. Transformation of the signal into the frequency domain is warranted by the combination of facts that (i) the system should be insensitive to the relative phases of the basis components and (ii) most of the important information is contained in lower order frequencies. Therefore, only the low order components of a frequency transform such as the cosine transform are required for the subsequent classification. These lower order terms are then fed into the final stage of the system, the classifier. The classifier is configured as a feed forward neural network whose weights are set by the well known back-propagation algorithm. The implementation of the classifier as a feed-forward neural network is supported by empirical observation that such networks exhibit an exceptional performance when cast with the task of pattern association. See for example [1].

3.1 Noise Reduction

The MEDN noise reducer is described in detail in the next section. A summary of its operation is as follows: A noise corrupted signal, $\widetilde{s}(\cdot)$, is input to the noise reducer. This corrupted signal is then convolved with a known window function, $g(\cdot)$. Equation 4.1 (see next section) is then implemented to recover the noise reduced signal, $s_{NR}(\cdot) \triangleq x_\lambda$.

3.2 Frequency Transform

Signals processed in a real system necessarily belong to the class of finite energy signals. If $x(\cdot)$ is a discrete time real signal this property can be described mathematically as $x \in \ell_2$, where

$$\ell_2 \triangleq \left\{ x : \|x\|_2^2 \triangleq \sum_{k=-\infty}^{\infty} x^2(k) < \infty \right\}. \tag{3.1}$$

In fact, the signals in ℓ_2 represent a superset of those that would be found in a real system since signals in a real system must be of finite length. Hence, it is clear that the type of signals we would like to analyze in a physical discrete system are finite length sequences. We will refer to this set of signals as $(\ell_2)_f$. Furthermore we would like to restrict ourselves to causal signals, i.e., those signals who are identically zero for all negative time. We denote this set, the projection of the space $(\ell_2)_f$ onto the space of all causal signals, as $C((\ell_2)_f)$. These sets are defined precisely below as

$$(\ell_2)_f \triangleq \left\{ x \in \ell_2 : \exists N \in \mathbb{N} \ni x(k) = 0 \ \forall k \ni |k| > N \right\} \tag{3.2}$$

$$C((\ell_2)_f) \triangleq \left\{ x \in \ell_2 : \exists N \in \mathbb{N} \ni x(k) = 0 \ \forall k \in \mathbb{N} \setminus \{0, 1, \cdots, N-1\} \right\} \tag{3.3}$$

and

$$C((\ell_2)_f) \subset (\ell_2)_f \subset \ell_2. \tag{3.4}$$

3.2.1 The Cosine Transform

An appropriate Fourier transform for signals in ℓ_2, and hence $(\ell_2)_f$, is the Discrete Time Fourier Transform (DTFT). Assuming that the discrete signal, $x \in \ell_2$, is generated

by sampling an analog signal where the sampling period is given as $T \in \mathbb{R}$, the continuous parameter of the DTFT is defined as $\theta = \omega T$. Here, ω represents frequency.

If, however, only signals in $C((\ell_2)_f)$ are of interest, the simple Cosine Transform may be employed. The continuous parameter of the Cosine Transform, θ, is defined as above in the case of the DTFT. A Cosine Transform has the desirable property that it is a mapping whose range is purely real. We will see that this property greatly facilitates classification. The Cosine Transform is defined below.

	Property	Description
1.	Linearity	$C\{ax(\cdot) + by(\cdot)\} = aC\{x(\cdot)\} + bC\{y(\cdot)\}$
2.	Real range	$\mathcal{R}\{C\{x(\cdot)\}\} = \mathbb{R}$

Table 1: *Cosine Transform Properties*

Suppose $x \in C((\ell_2)_f)$, then the Cosine Transform of $x(\cdot)$, denoted $C\{x(\cdot)\}$, is given as

$$C\{x(\cdot)\} \triangleq \sum_{k=0}^{N-1} x(k)\cos(k\theta), \tag{3.5}$$

where N is the finite length of the signal $x(\cdot)$. Some important properties of the Cosine Transform are presented below in Table 1. Let $x, y \in C((\ell_2)_f)$, and $a, b \in \mathbb{R}$. Property two in Table 1 suggests that the representation of signals in $C((\ell_2)_f)$ is greatly compacted from those in ℓ_2. Since the DTFT has complex range complete knowledge of both the phase and the magnitude of the transform is required; however, the Cosine Transform requires only knowledge of which half plane it is that the phase resides.

3.2.2 Computation of the Cosine Transform

The objective of real-time performance requires that each of the three components of the proposed system operate at comparable and *fast* speeds. At the rates of convergence (around a microsecond) for the MEDN and classifier networks, conventional frequency transformation methods are inadequate. For this reason, we again look to an analog approach [2].

Especially tractable for such an approach is the Discrete Hartley Transform (DHT) [3]. Defined in Equation 3.6, the DHT can be formulated as the matrix multiplication in 3.7. Let $\mathbf{x} \in \mathbb{R}^N$ representing a discrete signal of length N and define the cas function as $\text{cas}(t) \triangleq \sin(t) + \cos(t)$. Then, the DHT of $\mathbf{x}$, denoted $X_H(\theta)$ is

$$X_H(\theta) \triangleq \frac{1}{N}\sum_{k=0}^{N-1} x(k)\ \text{cas}(k\theta), \qquad \theta = \frac{2\pi n}{N} \tag{3.6}$$

OR

$$\tilde{\mathbf{X}} = D^{-1}\mathbf{x} \tag{3.7}$$

where $D = [D_{ij}]$ and $D_{ij} = \text{cas}(\frac{2\pi ij}{N})$. Among the many 'nice' properties of the matrix D are the facts that first, its inverse is given simply as $D^{-1} = \frac{1}{N}D$, and second, it has condition number equal to one.

We would now like to relate the DHT and the Cosine Transform. For notational convenience let the Cosine Transform be denoted as $X_C(\theta) \triangleq C\{x(\cdot)\}$. Noting that the cas function obeys the property

$$\text{cas}(\theta) + \text{cas}(-\theta) = 2\cos(\theta), \tag{3.8}$$

it is easy to see that the even part of the Hartley transform, $\mathcal{E}\{X_H(\theta)\}$, gives the Cosine Transform. That is

$$\mathcal{E}\{X_H(\theta)\} \triangleq \frac{1}{2}\{X_H(\theta) + X_H(-\theta)\} = X_C(\theta). \tag{3.9}$$

3.3 Classification

Classification as used here has the specific meaning of determining the vector, **a**, of weights that solves the problem (P). Although, classification in the sense of identifying an arbitrary parameter vector, Θ, for some input signal can be achieved to any desired accuracy by solving the problem (P). This is accomplished simply by partitioning the parameter space to the desired resolution. A pictorial representation of a possible partitioning is presented in Figure 3 for the case of a two dimensional parameter space, $\Theta = (\theta_1, \theta_2)$. Such a scheme, however, has the drawback that it dramatically increases the number of basis functions in the basis set $\mathcal{B}$.

Proposed for the classification is a feed forward three-layer pattern association neural network. Hence, the network consists of an input layer, an intermediate hidden layer and finally an output layer (see Figure 4). Trained in a supervisory manner, the associator network is presented with a training set of input/output patterns. Here the input pattern is a linear combination of the frequency transformed basis functions and the output pattern is the corresponding vector of weights. Training of the network is facilitated through the well known back-propagation algorithm [4].

Such neural topologies have been previously employed by many authors for the task of pattern association, e.g. [1] or [5]. For instance, Gorman and Sejnowski [1] in the classification of sonar signals. Gorman and Sejnowski essentially observed that as the number of hidden units was increased the performance of the classification was better for a moderate (more than 150) number of presentations of the training set. Specifically Gorman and Sejnowski obtained 99.8 percent classification accuracy with the ratio of hidden units to input units equal to $\frac{2}{3}$. For this reason the classifier was chosen to have the same number of hidden units as input units, i.e. a ratio of hidden units to input units equal to 1. The number of output units is simply the number of basis functions, N, in the basis set $\mathcal{B}$. Each level of activation of the output units then represents the

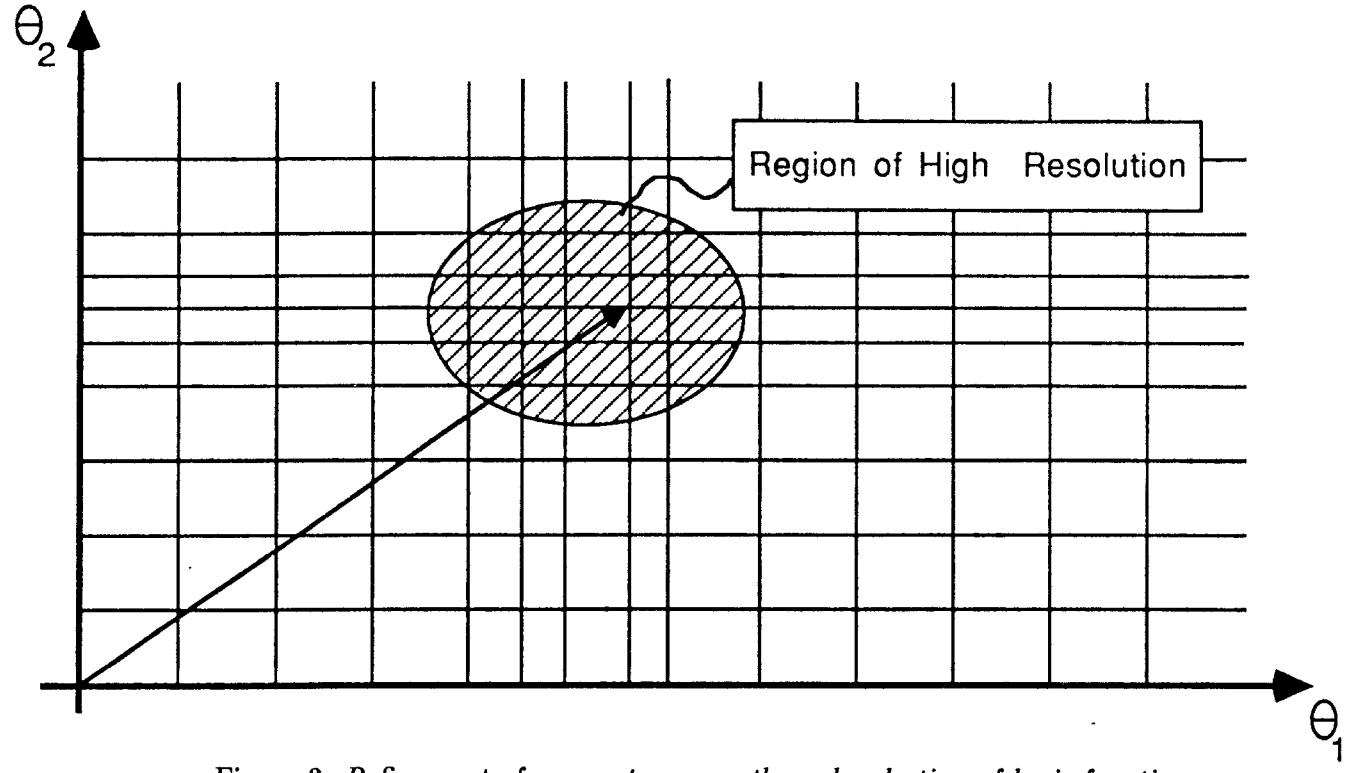


Figure 3: *Refinement of parameter space through selection of basis functions*

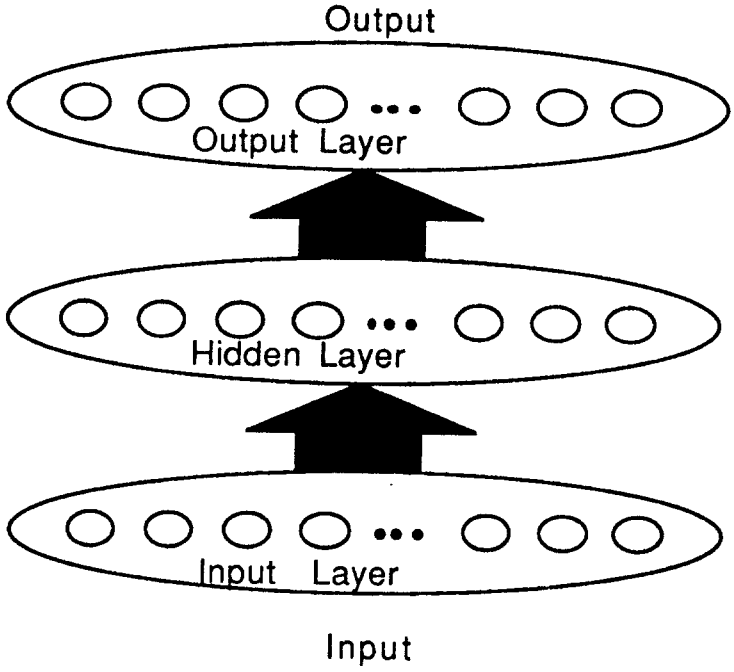


Figure 4: *Neural network classifier topology*

classifier's estimation of the weight. For ease of discussion the term 'network' will now be used in reference to this three layer topology just discussed unless otherwise indicated.

Simulations indicate that when the trained network is presented with an input outside the training set the result is linear interpolated, i.e. the network finds a linear fit for the data presented. Because of this linear interpolation property it is necessary that the transformation process be linear. Otherwise, classification of signals not in the training set will be unacceptable. Although the DTFT is linear, it has the drawback that its range is complex; and hence, the representation of one sample of the DTFT requires two real numbers. The Cosine Transform, which has purely real range, averts this drawback nicely.

4 The Deconvolution Network

Important to the operation of the system is the analog maximum entropy deconvolution network (MEDN) suggested by Marrian and Peckerar [6]. As its name suggests, this network's primary function is the task of deconvolving or deblurring a signal assumed previously convolved through some (perhaps physical) process. Because of this, it is not at all obvious why such a scheme is relevant to the solution of the problem discussed in Section 1. We defer motivation of the incorporation of the MEDN in the proposed system in lieu of a brief overview of the operation of the MEDN.

4.1 Operation of the MEDN

A detailed analysis of the MEDN is treated by Pati et. al. in [7]. Deconvolution or deblurring of signals in the presence of noise is in general ill-posed. Regularization is a technique to solve ill-posed problems in which *a priori* knowledge of the solution space is introduced into the solutions via a functional to be minimized (Poggio and Koch [8]). Letting x_λ denote the regularized solution to the deconvolution problem, we have,

$$x_\lambda = argmin\left\{\|y - Ax\|^2 + \lambda P(x)\right\} \tag{4.1}$$

Where it is assumed that the data has undergone a transformation of the form $y = Ax + \epsilon$, where A is a matrix representing the discretized convolution kernel, ϵ is a n-vector of noise components and x and y are both elements of n dimensional space.

In the particular case of the MEDN, the use of exponential amplifiers at the output nodes results in the introduction of a regularizing principle which is the negative of the Shannon entropy of the solution i.e. $P(x) = \sum_i x_i log(x_i)$. The constraints imposed by the entropy regularizer are smoothness and non-negativity of the solution. Hence, the entropy regularization "smooths" the solution resulting in reduction of the noise, i.e. an improvement in the signal to noise ratio.

By suitably adjusting the regularization weight λ, an arbitrary degree of smoothing can be achieved. However, if λ is allowed to be too large the resulting minimum will hardly approximate the true solution x. Therefore, there exists some optimal value for λ. Then, how should the parameter λ be chosen? Given that we would like the MEDN solution to minimize the mean square error (MSE), defined as

$$MSE = \|(x_\lambda - x)\|^2, \tag{4.2}$$

the use of optimization based design and the application of computer optimization software such as CONSOLE (Fan *et. al* [9]) seems appropriate.

4.2 Noise Reduction: Why the MEDN?

In the specific case of the radar return data as described earlier there is no convolution present in the noise model (i.e. the convolution kernel $A = I$). In fact, in general there is no reason to associate the operations of convolution and noise reduction in any way. This prompts the question of the appropriateness of a deconvolution network for the task of noise reduction.

Measurement of the noise reduction is based on a squared error sum methodology. Knowledge of the uncorrupted signal, s, is assumed for the computation. The magnitude of the degradation of the signal is then measured in deterministic terms. Given the degraded signal $\widehat{s}$, the degradation, $d(\widehat{s})$, of the signal is

$$d(\widehat{s}) \triangleq \sum_{t=0}^{N-1} (\widehat{s}(t) - s(t))^2. \tag{4.3}$$

Note that the relationship between the signal to noise ratio and $d(\cdot)$ is on the average a monotonically increasing one. That is to say that on average the larger the signal to noise ratio, the larger $d(\cdot)$ will be for a particular realization of $s(\cdot)$ through noise.

Noise reduction with respect to the MEDN is then simply given as the ratio of the noise degradation functions of the signals present at the output and input of the MEDN respectively. If s_{input} and s_{output} represent the input and output of the MEDN, the noise reduction, NR, is given as

$$NR = \frac{d(s_{output})}{d(s_{input})}. \tag{4.4}$$

Clearly a value for $NR < 1$ is required of the MEDN.

It is obvious, as alluded to above, that deconvolution per se is unwarranted in the problem (P). What is of importance, however, is the regularizing principle employed by the MEDN. Through exploitation of the regularizing properties of the network, it is possible to achieve the desired end of noise reduction. This claim is supported through the observations that the MEDN

1. exhibits non-linear reduction in the noise (see Figure 3)

2. is demonstrably better than simple low pass (see Figure 5).

4.3 Implementation of the MEDN Noise Reducer

Since there is no convolution present in the signal, $\widetilde{s}(\cdot)$, it is necessary to convolve the noisy data with a non-trivial convolution kernel. The reason for this is that if the convolution kernel were trivial, the MEDN would become decoupled. That is, the solution given by the vector equation 4.1 for a given element would become independent of the

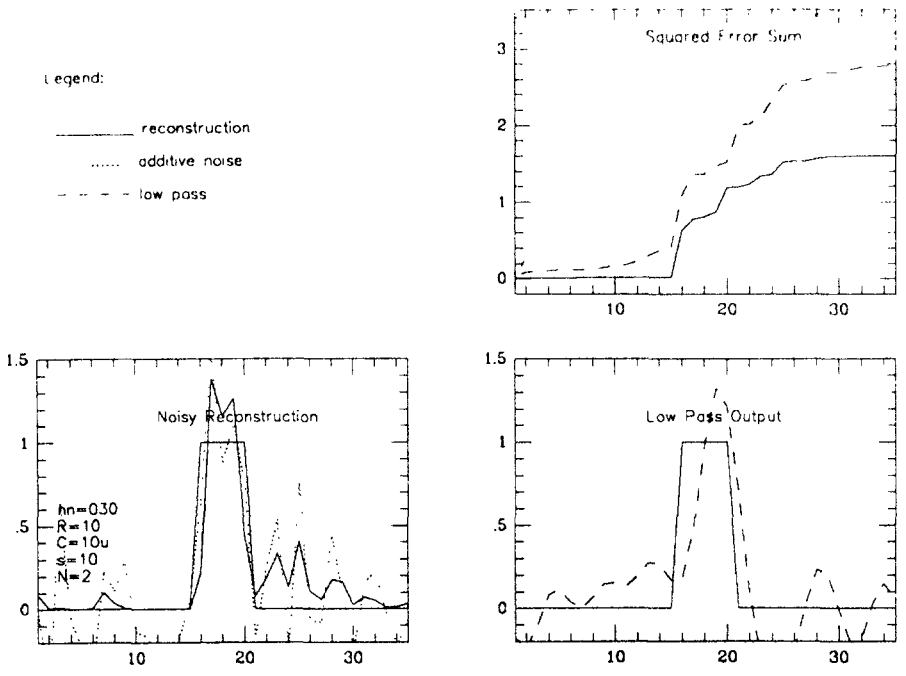


Figure 5: *Noise reduction using MEDN as compared to simple low pass*

value of its neighboring elements. Or equivalently, the vector equation 4.1 becomes n independent scalar equations and there is no hope of smoothing the data. To avert this problem, the incoming data is preconvolved with a non-trivial convolution kernel or window function, denoted by $g(\cdot)$ and given in Equation 4.5.

$$g(t) = \begin{cases} h, & \text{if } |t| < N_k \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

Thus, the pulse width of the kernel, $2N_k - 1$, is an added dimension to the space of parameters over which the minimization of 4.2 will range.

5 System Simulation

Extensive computer simulations of the proposed system have been performed. As a test case, the value $N = 2$ for the number of basis functions in the basis set, $\mathcal{B}$ was chosen in the simulations. Presented in Table 2 below are other choices for system parameters.

The first block of the proposed architecture, the MEDN noise reducer, has been simulated using the interactive simulation package SIMNON (Lund Institute Of Technology). Appendix A contains typical simulation results. The dash lined functions represent the respective noise functions in both the Squared Error, and Noisy Reconstruction graphs and the solid line represents the spectrum of the uncorrupted data in the Transform graph. Parameter *nn* in the figures indicates the standard deviation of the noise which is added to the normalized input signal. This value of *nn* is obtained by multiplying the standard deviation by a factor of 100 followed by truncation, i.e., $nn = int(100\sigma)$.

Symbol	Parameter	Value
	Convolver	
h	kernel height	1
	MEDN	
R	Resistance	10 Ω
C	Capacitance	10 μF
S	Feedback Gain	10
N_k	Kernel Width	5
	Classifier	
n_i	Number of input units	11
n_h	Number of hidden units	11
n_o	Number of output units	2

Table 2: *System Parameters*

Inspection of the Squared Error graphs indicates that substantial noise reduction has been achieved as defined by equation 4.4.

As discussed earlier, the transform stage of the system is implemented as a simple Cosine Transform. This simply entails evaluating equation 3.5 for $\theta = \frac{\pi}{n}k$, $k = 0, 1, \ldots, n_i - 1$ (=10) where n is the length of the input signal. Typical transforms can be seen in Appendix A.

Simulation of the back-propagation classification network is supported by the neural network software package written by Rumelhart and McClelland [4]. Normalization of the elements of the training set greatly reduces its size since only parameter vectors whose elements add to one need be included. This effectively reduces the output space to a dimension one less than the original space. For the simulations in Appendix A, the network was trained on a set of only size ten.

A summary of the performance of the simulation of the system is presented in Table 3. The table consists of five sets of runs; each run is associated with a different signal to noise ratio, S/N. Each run set consists of five sets of parameters from the training set and five outside the training set. The values a_1 and a_2 represent the respective weights of the basis functions composing the deterministic portion of the input signal. Classification results are displayed in the following two columns as $\widetilde{a}_1$ and $\widetilde{a}_2$ corresponding to the two respective estimates of the basis weights, a_1 and a_2. The final four columns of the table present various percent error statistics: γ_1 and γ_2 are the respective percent error in the estimates of a_1 and a_2, $\bar{\gamma}$ is the average percent error estimate for both a_1 and a_2, and finally $\bar{\gamma}_T$ is the total average error estimate for all runs in the test set.

This simulation data seems to suggest an inverse relationship between the average total error, $\bar{\gamma}_T$ and the signal to noise ratio, S/N. That is the average total error is a monotonically decreasing function of the signal to noise ratio. Moreover, we have fitted a curve of the form

$$f(x) = b + e^{(\mu - \alpha x)} \quad , \quad x \in [0, \infty) \tag{5.1}$$

S/N	a_1	a_2	$\widetilde{a}_1$	$\widetilde{a}_2$	γ_1	γ_2	$\bar{\gamma}$	$\bar{\gamma}_T$
	0.2	0.8	0.15	0.85	26.96	6.72	16.84	
	0.4	0.6	0.30	0.70	23.99	15.96	19.97	
	0.5	0.5	0.42	0.58	16.23	16.22	16.23	
	0.6	0.4	0.54	0.46	9.53	14.31	11.92	
	0.8	0.2	0.76	0.24	5.13	20.61	12.87	
0.0 dB	0.1	0.5	0.08	0.52	22.62	4.51	13.57	14.43
	0.4	0.5	0.32	0.58	20.63	16.48	18.55	
	0.5	0.3	0.46	0.34	8.27	13.82	11.05	
	0.8	0.4	0.75	0.45	6.60	13.25	9.93	
	0.9	0.7	0.79	0.81	11.73	15.09	13.41	
	0.2	0.8	0.16	0.84	21.20	5.25	13.23	
	0.4	0.6	0.32	0.68	19.89	13.20	16.54	
	0.5	0.5	0.44	0.56	12.77	12.77	12.77	
	0.6	0.4	0.56	0.44	6.90	10.40	8.65	
	0.8	0.2	0.77	0.23	3.87	15.59	9.73	
1.0 dB	0.1	0.5	0.08	0.52	16.27	3.21	9.74	11.04
	0.4	0.5	0.33	0.57	16.83	13.42	15.13	
	0.5	0.3	0.47	0.33	5.82	9.76	7.79	
	0.8	0.4	0.76	0.44	4.44	8.97	6.71	
	0.9	0.7	0.82	0.78	8.81	11.36	10.09	
	0.2	0.8	0.17	0.83	16.52	4.07	10.30	
	0.4	0.6	0.33	0.67	16.71	11.09	13.90	
	0.5	0.5	0.45	0.55	10.13	10.14	10.13	
	0.6	0.4	0.57	0.43	4.84	7.34	6.09	
	0.8	0.2	0.78	0.22	2.92	11.79	7.36	
2.0 dB	0.1	0.5	0.09	0.51	11.02	2.15	6.59	8.38
	0.4	0.5	0.34	0.56	13.88	11.07	12.48	
	0.5	0.3	0.48	0.32	3.92	6.61	5.27	
	0.8	0.4	0.78	0.42	2.78	5.67	4.22	
	0.9	0.7	0.84	0.76	6.54	8.46	7.50	
	0.2	0.8	0.17	0.83	12.78	3.14	7.96	
	0.4	0.6	0.34	0.66	14.15	9.38	11.77	
	0.5	0.5	0.46	0.54	8.03	8.05	8.04	
	0.6	0.4	0.58	0.42	3.24	4.95	4.09	
	0.8	0.2	0.78	0.22	2.19	8.89	5.54	
3.0 dB	0.1	0.5	0.09	0.51	6.86	1.32	4.09	6.29
	0.4	0.5	0.35	0.55	11.52	9.19	10.36	
	0.5	0.3	0.49	0.31	2.43	4.16	3.29	
	0.8	0.4	0.79	0.41	1.49	3.10	2.30	
	0.9	0.7	0.86	0.74	4.79	6.22	5.51	
	0.2	0.8	0.21	0.79	2.57	0.69	1.63	
	0.4	0.6	0.38	0.62	4.29	2.85	3.57	
	0.5	0.5	0.50	0.50	0.34	0.39	0.37	
	0.6	0.4	0.61	0.39	2.43	3.53	2.98	
	0.8	0.2	0.79	0.21	0.91	3.49	2.20	
∞ dB	0.1	0.5	0.11	0.49	10.40	2.12	6.26	2.93
	0.4	0.5	0.39	0.51	2.56	2.06	2.31	
	0.5	0.3	0.51	0.29	2.81	4.53	3.67	
	0.8	0.4	0.82	0.38	2.98	5.79	4.39	
	0.9	0.7	0.91	0.69	1.60	1.96	1.78	

Table 3: *Simulation Summary*

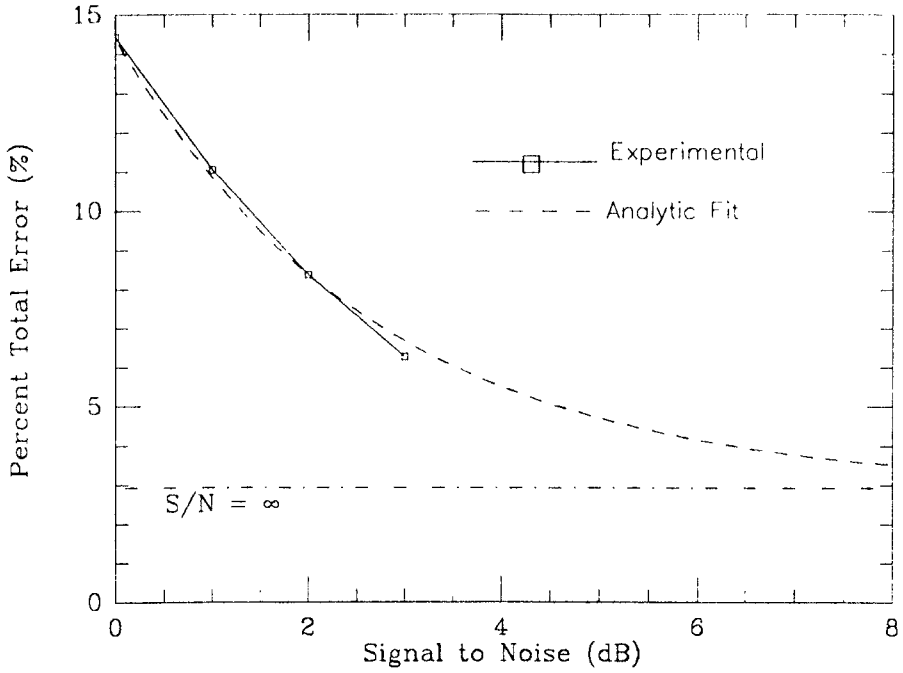


Figure 6: *The average total error of parameter estimation as a function of the signal to noise ratio.*

to the experimental data. Figure 6 displays the experimental and fitted (analytical) curves obtained. The values obtained for the parameters are $b = 2.93, \mu = 2.44$, and $\alpha = 0.372$.

6 VLSI Implementation

In order to realize the full computational power of neural networks it is important to consider implementation of neural networks as analog electrical circuits. Although the current state of VLSI technology does not allow for implementation of extremely large networks, there is still a class of useful networks that can be implemented. The networks we consider here are among those for which currently implementable sizes are sufficient to prove useful. As discussed earlier, foremost amongst our objectives, is the design and implementation of a system suitable for practical applications. Essential to the practicality of such a system are:

1. The ability to perform the desired decomposition and classification within the constraints of a demanding real-time environment.

2. Physical compactness and modest power consumption, so as to satisfy constraints of the physical environment (e.g. in missile guidance applications).

In this section we discuss analog VLSI implementation of the proposed system for radar signal decomposition and classification. As we shall show, analog VLSI implementation results in a system capable of satisfying the desired performance criteria.

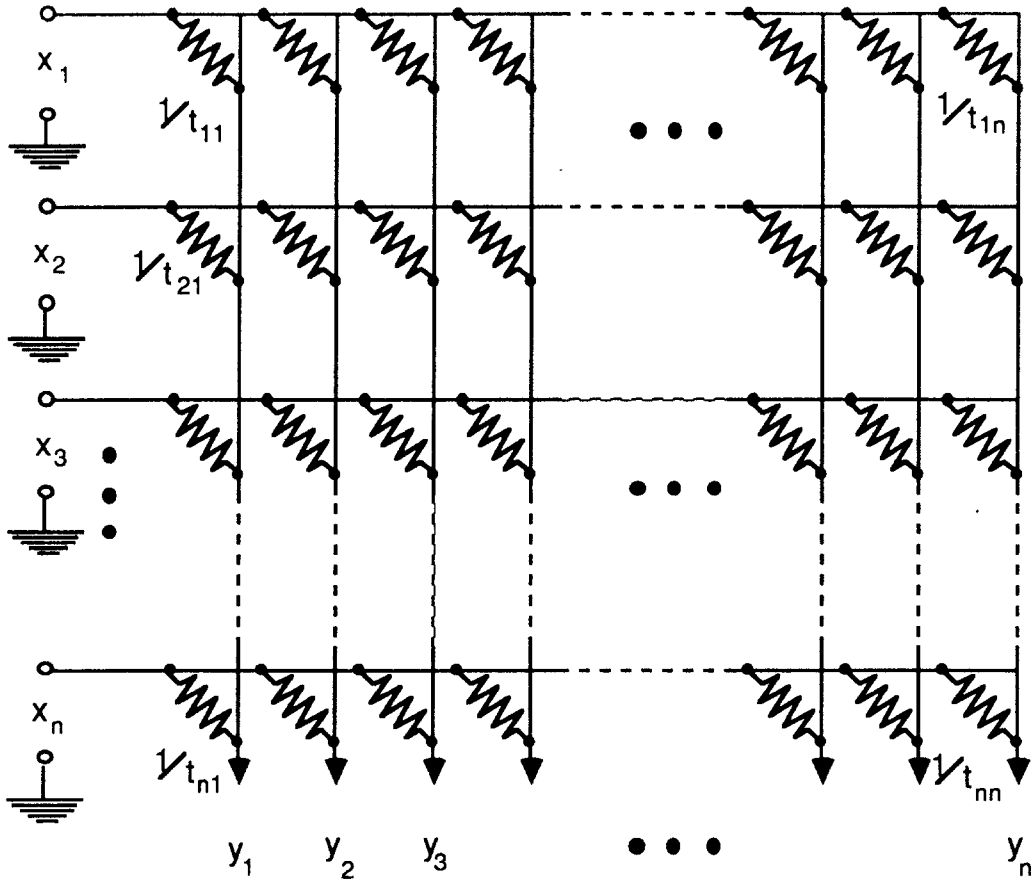


Figure 7: Network to implement discretized analog convolution

6.1 The Convolution Subnetwork

Given a vector $x \in \mathbb{R}^n$ of sampled data, convolution with a given convolution kernel is described by the system of equations, $y = Tx$, where $y, x \in \mathbb{R}^n$, and $T \in \mathbb{R}^{n \times n}$ is the matrix representation of the discretized convolution kernel. Convolution is easily implemented in analog circuitry by the network shown in Figure 7. Inputs to the network in Figure 7 are the voltages $x_1, \ldots, x_n$. Connections within the network are made by resistors with values $R_{ij} = 1/t_{ij}$. Thus the contribution to the current at the jth output y_j due to x_i is given by Ohm's Law to be x_i/R_{ij}. Hence the output currents are given by

$$y_j = \sum_{i=1}^{N} \frac{x_i}{R_{ij}} = \sum_{i=1}^{N} t_{ij} x_i \qquad j = 1, \ldots, N. \tag{6.1}$$

Which implements the desired convolution.

For the particular 'boxcar' convolution kernel given by Equation 4.5, the matrix $T = [t_{ij}]$ is a band-diagonal matrix given by,

$$t_{ij} = \begin{cases} h & \text{if } |i-j| \leq (N_k - 1)/2 \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

Thus the number of nonzero connections required is $n + 2\sum_{i=1}^{(N_k-1)/2}(n-i)$. Fabrication of the required network interconnections is thus facilitated by choice of the boxcar

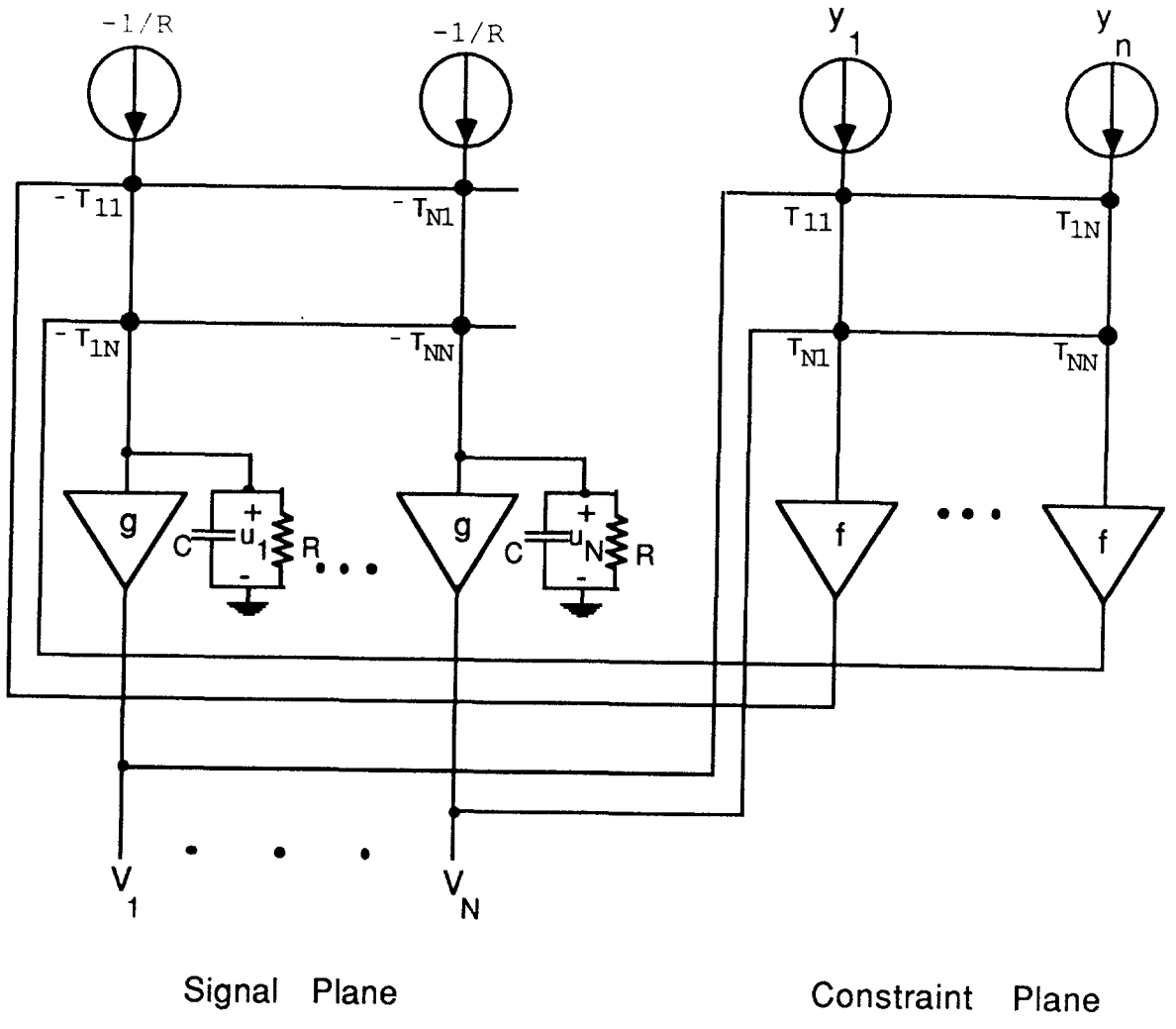


Figure 8: *The Maximum Entropy Deconvolution Network (MEDN)*

convolution kernel, since

1. only a small number of connections are required in comparison to the number of possible connections (n^2) for an arbitrary choice of the convolution kernel, and
2. since the weights (resistance values) for the connections are all identical, errors due to processing induced variations in the circuit are minimized.

6.2 The Deconvolution Subnetwork

As discussed earlier, regularized deconvolution is performed by the maximum entropy deconvolution network (MEDN) shown schematically in Figure 8. Inputs to the MEDN $(y_1, \ldots, y_n)$ are currents which are provided by the outputs of the convolution network just described. An initial attempt at integrated circuit implementation of the deconvolution network has been completed, but remains untested. In addition, two prototype breadboard networks have been constructed and have been used to study performance of the deconvolution network in terms of speed of convergence and accuracy of solutions.

Accurate assessments of convergence time for the network are not easily made using digital computer simulations. Also, in analysis of the deconvolution network (see [7]) it was assumed that any dynamics associated with the constraint plane could be ignored provided that the signal plane amplifiers are sufficiently slower in response. In practical implementations of such a network, it is necessary to understand what effects delays in the constraint plane response may have upon the network. It is ultimately the constraint plane dynamics which limit the speed of convergence which is achievable. A formal treatment of this subject is to be found in Marcus and Westervelt [10].

6.2.1 Breadboard Prototype Deconvolution Network

A prototype breadboard model of the deconvolution network was constructed using 'off-the-shelf' operational amplifiers, resistors and capacitors. The network was constructed with seven signal plane nodes and seven constraint plane nodes. Since the purpose of constructing the breadboard prototype was to estimate the speed of convergence achievable by such a network, exponential amplifiers in the signal plane were replaced by unity gain linear amplifiers to simplify the circuit [2]. Replacing the exponential amplifiers by linear amplifiers results in the entropy regularizer being replaced by a regularizer of the form,

$$\lambda M(v) = \frac{1}{R}\sum_i v_i^2. \tag{6.3}$$

In Figure 9 a circuit diagram of a single signal plane node is shown. Each signal plane node consists of two stages of amplification. Associated with the first stage is the feedback capacitor C, which introduces the relevant network dynamics, and the feedback resistor R, which introduces and weights ($\lambda = 1/R$) the regularizing term in the energy function. The second stage of each signal plane node is configured as an analog inverter. Negative connection weights are implemented simply by using the inverted output of the node. Although negative connection weights are not required in the particular application we consider here, they were necessary for the tactile sensing application (see [7]) in which the network was tested. Figure 10 shows a single constraint plane node. Each constraint plane node also consists of two stages. The first stage is configured as a virtual ground transimpedance amplifier, which provides the feedback gain for signals fed back to the signal plane. As in the signal plane, the second stage of each constraint plane node is an analog inverter.

In Figure 11 photographs of the 7-Channel breadboard prototype of the deconvolution net work and the experimental setup used for testing it are shown. Input currents to the network are clocked using a 1 kHz relaxation oscillator so as observe transients (as the network evolves) using an oscilloscope. Outputs of the network are captured by a MetaResearch data acquisition board used in conjunction with a Macintosh Plus computer and the resultant reconstruction is plotted on the Macintosh display.

The rise time of the constraint plane amplifiers was measured to be approximately 1 μsec. Actual response time of the constraint plane would be longer than this since the parallel combination of all resistors connected to the input of any node contribute to the RC time constant. It was observed that for choices of the signal plane capacitors C for

[2] A second breadboard prototype was also constructed which contained the exponential amplifiers, but was used to solve a different problem (see [11]).

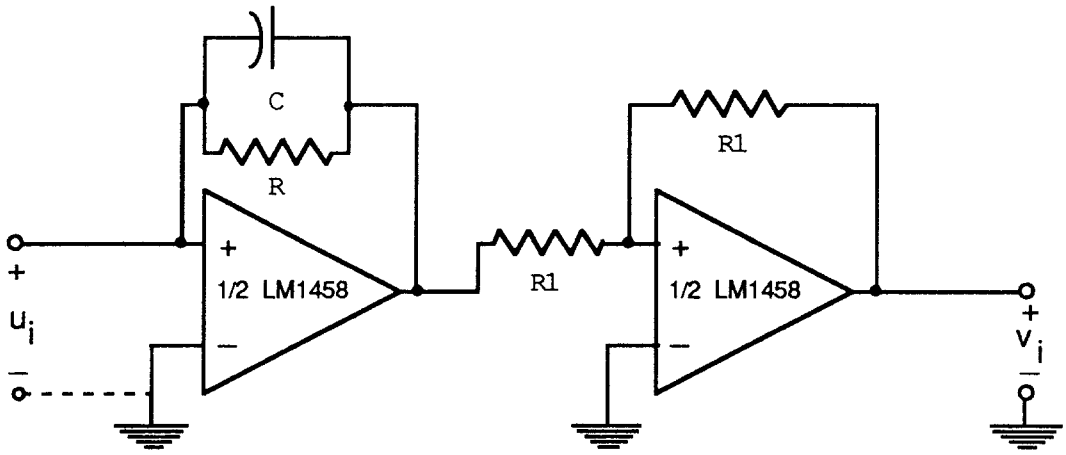


Figure 9: *Schematic circuit diagram of a single signal plane node*

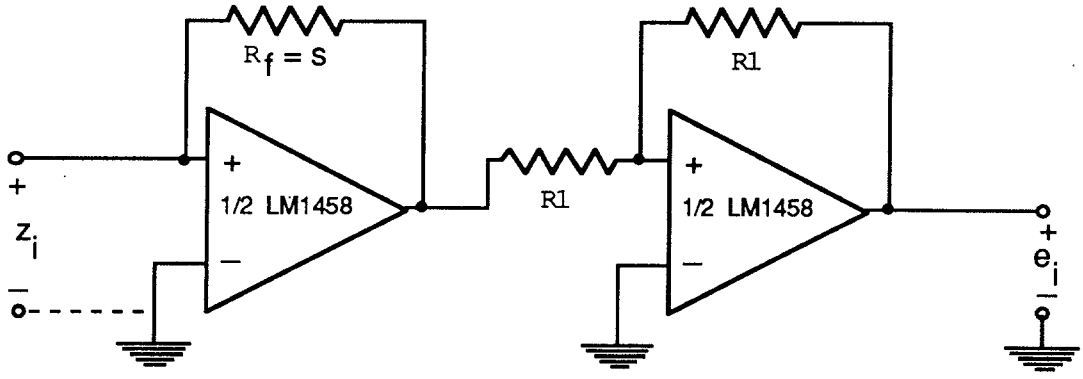


Figure 10: *Schematic circuit diagram of a single constraint plane node*

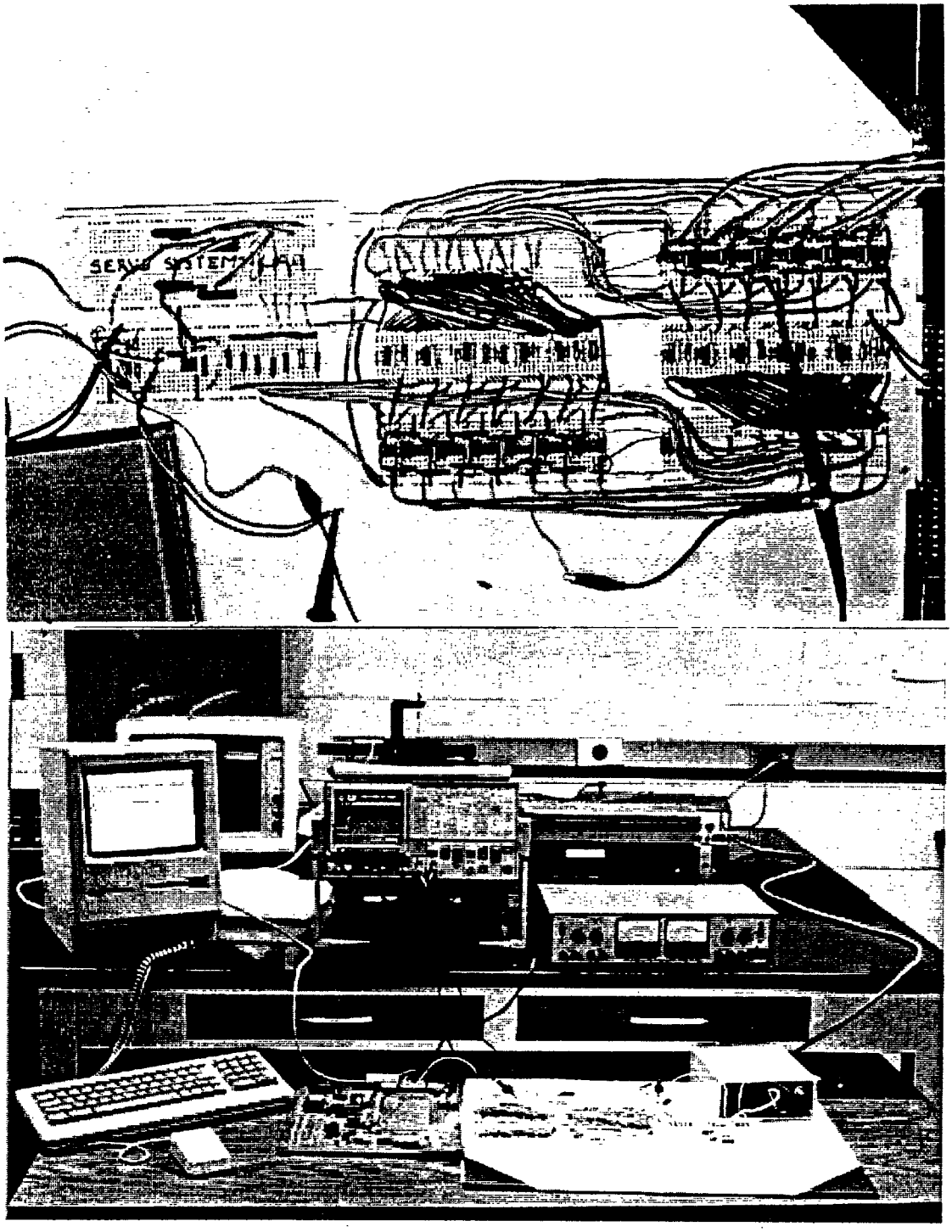


Figure 11: *Top: Photograph of 7-Channel breadboard prototype of deconvolution network Bottom: Photograph of experimental setup used to test the deconvolution network.*

which the rise time of the outputs of the network would be below 10 μsec, the outputs would oscillate i.e. the network was unstable. For C=10 pF the rise time of the outputs of the network was measured to be 10 μsec (see Figure 12). It is clear that the use of faster operational amplifiers would result in an increase in achievable speed since this would decrease the constraint plane response time and thereby permit a decrease in the time constant of the signal plane.

Settling time and overshoot of the outputs of the network are controlled by the gain of the constraint plane nodes. CONSOLE, a CAD tool for parametric optimization of dynamical systems, (see [9]) was used to choose a value for the gain so as to minimize overshoot and settling time.

6.2.2 Integrated Circuit Prototype of Deconvolution Network

A prototype analog integrated circuit implementation of the deconvolution network described here has been fabricated, but remains to be tested. A hierarchical design philosophy is practiced in this initial implementation. The deconvolution network may be thought of as composed of two sections: (i) Active components of the network including signal and constraint plane amplifiers and (ii) The functionally passive[3] resistive interconnection matrix. These two sections may also be thought of in the following manner. Once the size of the deconvolution network (number of inputs and outputs) has been decided, the amplifiers of the network are determined. However, the resistive matrix may be a variable entity. Thus the network may be thought of as being composed of a fixed part and a variable part.

If fixed resistors are to be used to implement the interconnect matrix, then some provision should be made to change this matrix without having to refabricate the rest of the network. In order to provide some flexibility in the choice of the interconnect matrix and to permit the use of two different fabrication technologies, the deconvolution network was fabricated as two separate integrated circuits.

The amplifier chip is designed to serve as the 'motherboard' for the network on top of which the resistive connection matrix chip is placed (see Figure 13). Connections between the two chips are made by local wire bonds between bonding pads provided for this purpose on both chips. This approach also facilitates experimentation with different types of connection matrices such as those with programmable connections.

The Amplifier Chip Figure 14 shows the layout of the amplifier section of the deconvolution network. This chip provides the signal plane and constraint plane amplifiers for a deconvolution network with eleven input nodes and eleven output nodes. Fabrication of the amplifier section of the deconvolution network was undertaken through the MOSIS facility, using a two metal layer CMOS p-well technology with a minimum feature size of 3μm. The size of the amplifier chip is approximately $6300\mu m \times 8800\mu m$. In the center of the amplifier chip, a $4000\mu m \times 3200\mu m$ space has been provided for placement of the second chip containing the resistive connection matrix. It is actually not necessary to

[3]The term 'functionally passive' here is used to describe the fact that in some situations it is desirable to use active circuit components configured to look like a passive resistor from an input/output standpoint.

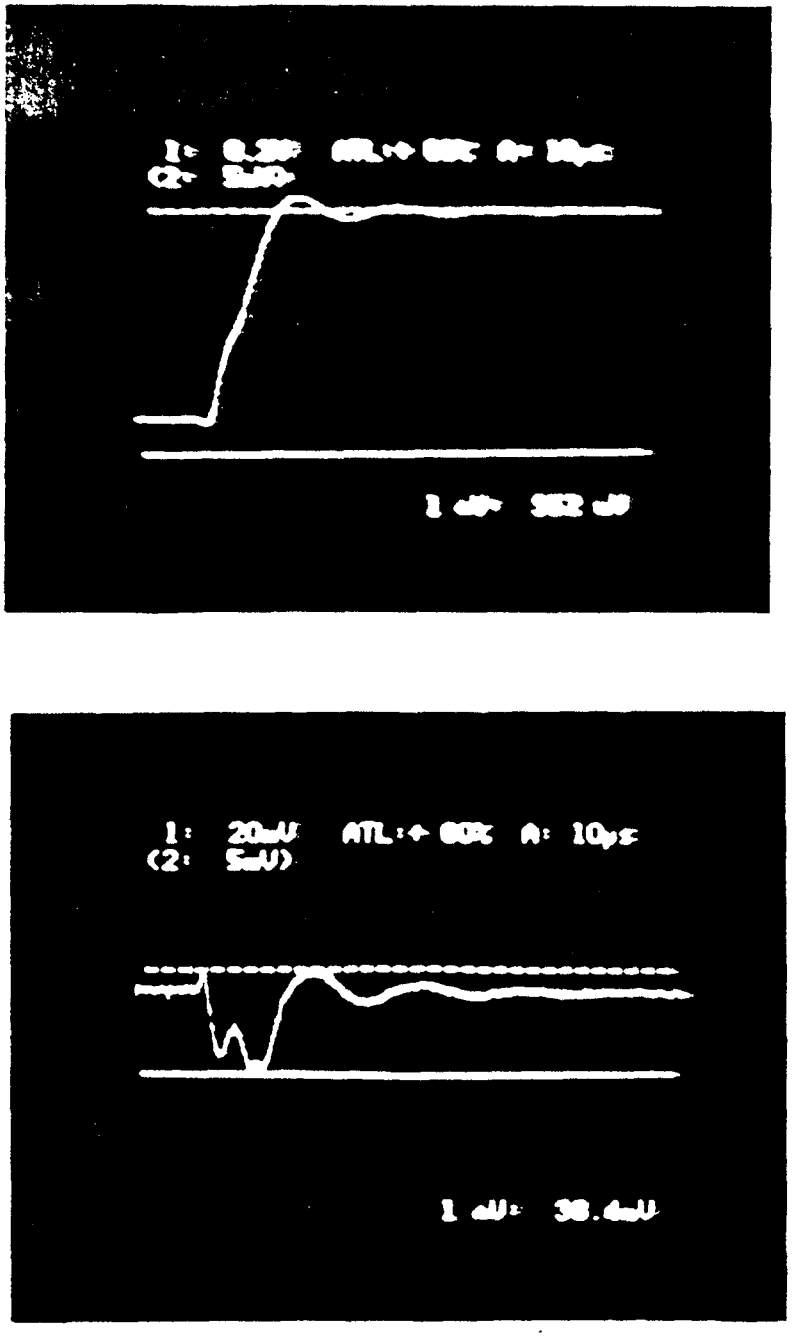


Figure 12: *Photographs of oscilloscope traces showing, Top: time evolution of a single output of the signal plane, and Bottom: time evolution of a single output of the constraint plane, for the 7-channel breadboard prototype. deconvolution network.*

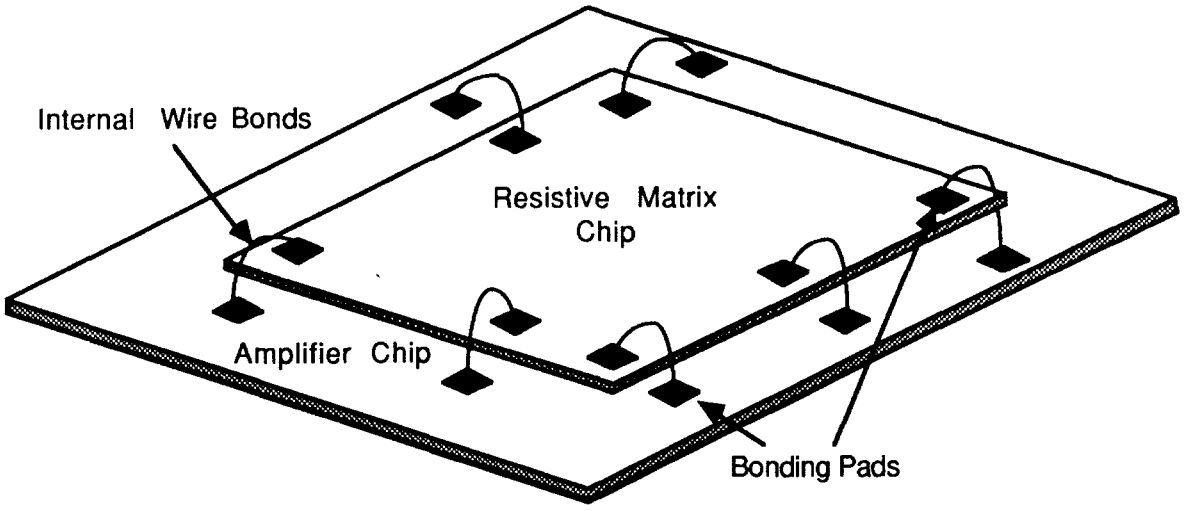


Figure 13: *Resistive matrix chip is placed directly atop the amplifier chip and connected to it using local bonding*

leave this space since the chip is passivated, but for an initial implementation, it was provided.

A total of forty four operational amplifiers are implemented on this chip; twenty two for the signal plane and twenty two for the constraint plane. Each signal plane node and constraint plane node is identical in structure to the signal and constraint plane nodes used for the breadboard prototype. Both the inverted and noninverted outputs of the signal and constraint plane nodes are connected to bonding pads adjacent to the area where the resistive matrix is to be placed. Thus provision is made for the implementation of positive and negative connection weights. Outputs and inputs of both the signal plane nodes and constraint plane nodes are connected to bonding pads located on the periphery of the die to permit access to these nodes after the chip has been packaged. Packaging of the chip requires a package with a minimum of fifty pins, if the inputs and outputs all nodes are to be accessible.

The basic building block of the amplifier chip is the operational amplifier. Figure 16 shows a circuit schematic of the operational amplifier designed for this purpose. Design of the amplifier takes into consideration the current driving requirements for use in the deconvolution network. Inputs on each amplifier are diode protected against spikes. A source follower output stage is used on every amplifier so as to meet the current driving requirements while maintaining relatively low circuit complexity (compared to, for example a push-pull output stage).

As can be seen from Figure 16, an internal compensation capacitor of 5pf is required for every operational amplifier. In addition to this, external feedback capacitors of about 10pF are required at the first stage of every signal plane node. Since the technology

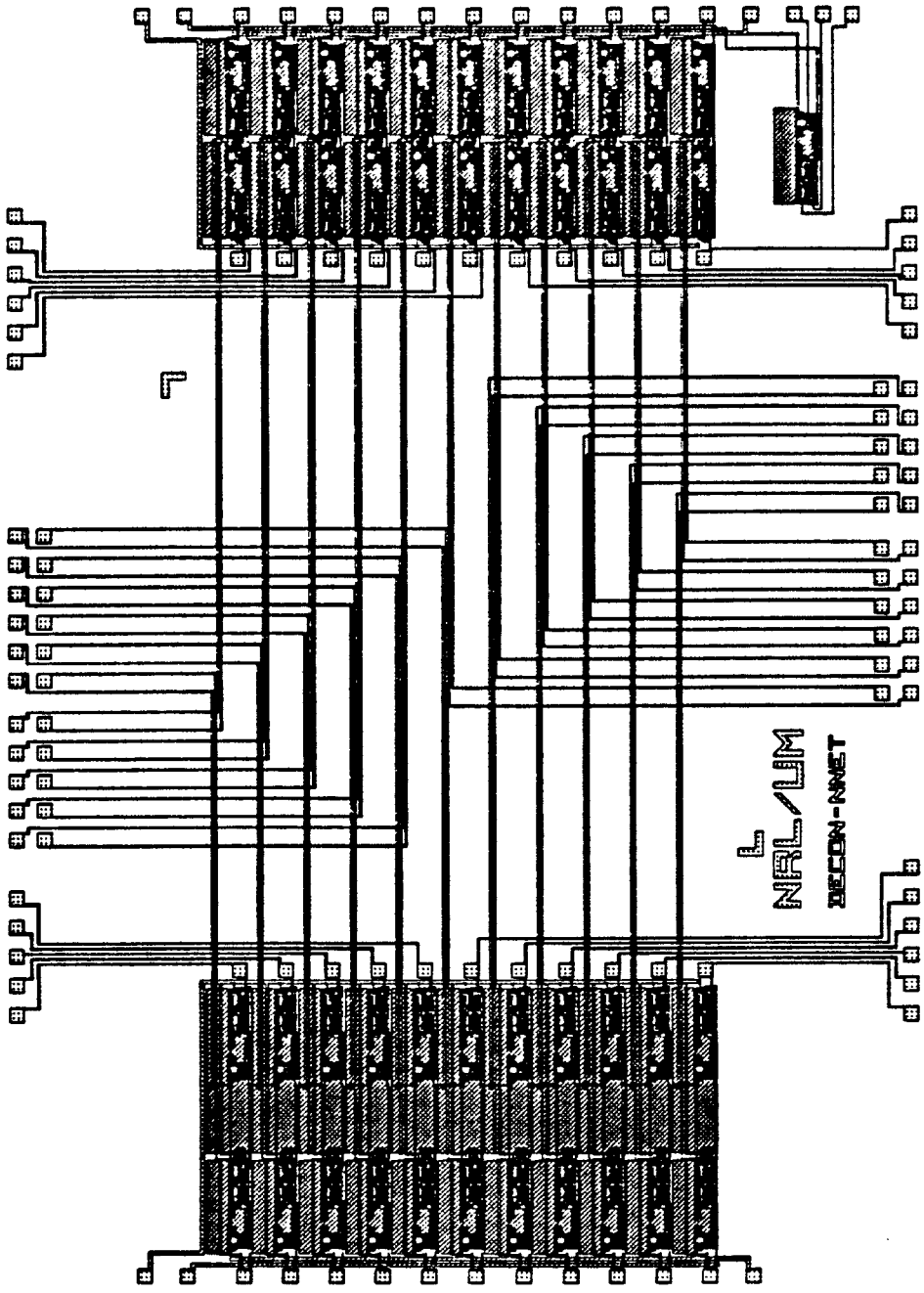


Figure 14: *Layout of integrated circuit chip containing all amplifiers for an eleven channel deconvolution network*

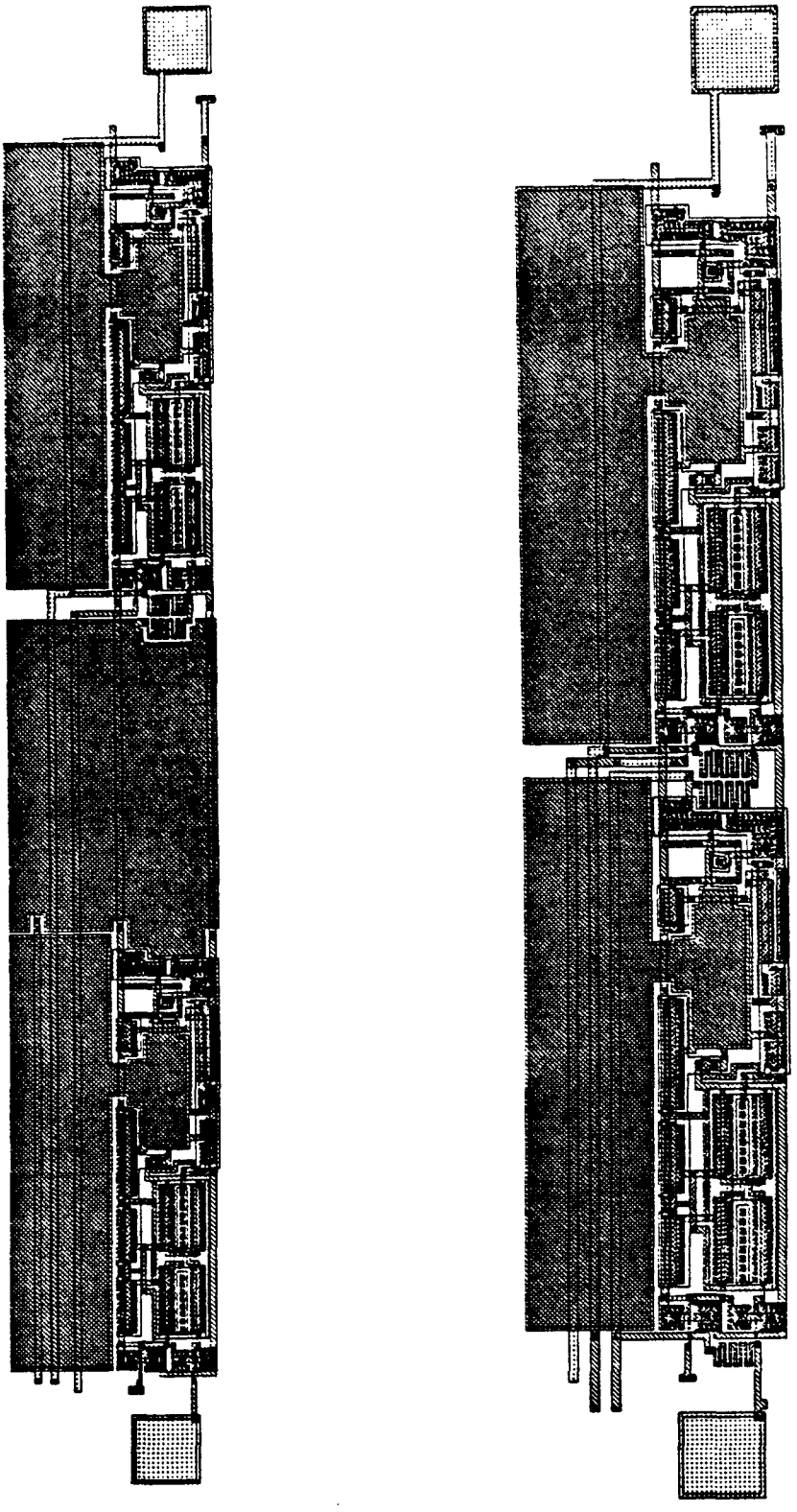


Figure 15: *Left: Layout of signal plane amplifier; Right: Layout of constraint plane amplifier.*

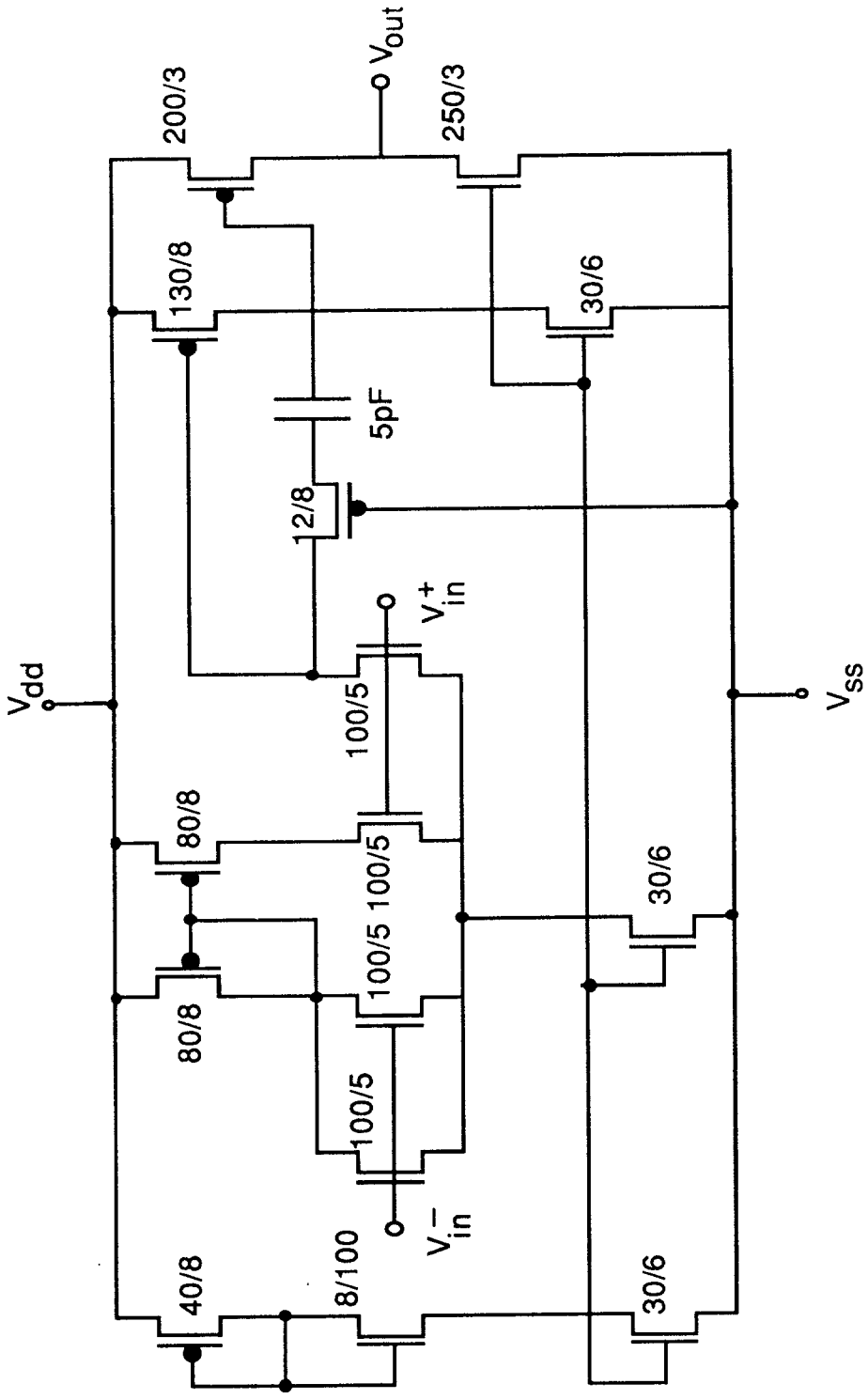


Figure 16: *Circuit diagram of operational amplifiers designed for integrated circuit implementation of deconvolution network*

available did not include an additional electrode layer (i.e. a second layer of polysilicon to be used in forming capacitors), all capacitors in the network are formed as parallel plate capacitors with polysilicon as one electrode and the first metal layer as the second electrode. Capacitance between the first metal layer and polysilicon is approximately a factor of eight less than the capacitance between polysilicon and an electrode layer. Hence about eight times as much area is used to form the capacitors, compared to capacitors formed using an electrode layer. Since about half of the area used by constraint plane nodes and two thirds of the area used by the signal plane nodes is used to form the capacitors, about seven twelfths of the total area used by the amplifiers is taken up by capacitors. Hence the use of a separate electrode layer would result in about a 50% reduction in space utilization. Therefore a network with twenty two inputs and twenty two outputs is a trivial extension of the current structure (which includes the space provided for the resistive matrix) using a comparable $3\mu m$ technology.

The Resistive Interconnection Matrix Chip In any implementation of a matrix of resistive elements, it is necessary to consider the effects of process induced variations on the performance of the network with which it is to be used. Current integrated circuit process technology may introduce variations in the value of any given resistor as large as 20%. It must either be established that degradation in performance of the network due to such variations is irrelevant, or an approach to implementation must be taken which preserves the essential characteristics of the interconnection matrix in the face of process induced variations. The latter approach was taken in this implementation.

Since in the case of the deconvolution network, the resistive interconnection matrix is a discretization of a convolution kernel, it is reasonable to try to preserve the shape of the connectivity profile. That is, it is the ratiometric relationship of the connective weights that is truely important since any other variations correspond to a simple scaling of the inputs or outputs. For the boxcar convolution kernel considered here preservation of the shape of the connectivity profile corresponds to the requirement that all nonzero connections should be equal. In an attempt to preserve the shape of the connectivity profile in the face of processing induced variations, the connective weights are quantized into quanta of discrete resistive elements. Figure 18 shows the layout of the resistive matrix. A discrete resistive element is formed by etching a $5\mu m \times 5\mu m$ opening in the oxide layer between two metal wires and then evaporating silicon into the opening (see Figure 17). Amorphous silicon, therefore forms the resistive material. Each such discrete resistor has a resistance value that depends on the thickness of the oxide layer and the area of the opening. For the oxide thickness used (and $5\mu m \times 5\mu m$ openings) the value of a single discrete resistor was measured to be approximately $35K\Omega$. A wire grid is formed using the first metal layer to construct parallel wires in one direction and the second metal layer to construct wires in a direction orthogonal to the wires formed in the first metal layer. At each point on the grid, a connective weight is defined by placing a number of discrete resistors of the type described at that point. The strength of connection at any point on the grid is determined by the number of discrete resistors placed there. Ratiometric relationships between elements of the connection matrix should for the most part be preserved in this approach since the ratio of two connective weights is primarily determined by the ratio of the number of discrete resistors forming each connection Any variations in the ratios are the effects of phenomena such as nonuniform oxide thickness over the area of the die, nonuniform deposition of amorphous silicon which are relatively

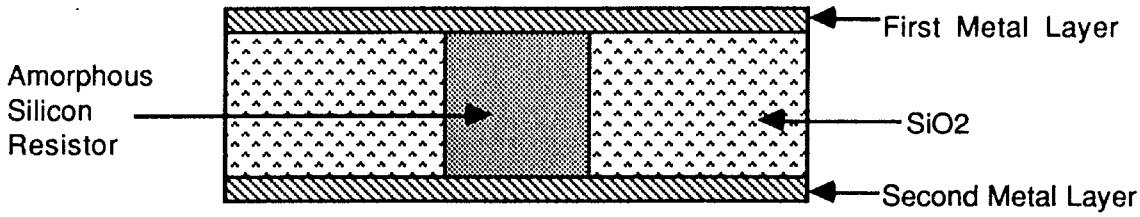


Figure 17: *A discrete resistive element as used in the resistive matrix chip.*

small effects.

6.3 Cosine Transformation in VLSI

As discussed earlier, the Cosine Transform of the sampled signal is necessary in order to perform the task of classification. It is observed that that the Discrete Hartley Transform (DHT) of a vector x of samples of a signal may be obtained as the result of multiplication by the inverse of the self-adjoint matrix D, i.e.

$$\tilde{X} = D^{-1}x = \frac{1}{N}Dx, \tag{6.4}$$

where $\tilde{X}$ is the Discrete Hartley transform of x. It is also observed that the Cosine Transform may be obtained by a simple average of the components of the DHT (see equation 3.9).

In [2] one approach to implementing the DHT using an analog network is discussed. However, the approach taken in [2] involves a *recurrent* network in which dynamics govern the speed of convergence as in the MEDN. Also the implementation in [2] requires the implementation of a large number of amplifiers. We consider a simpler approach here in which we perform the vector matrix multiplication required for the DHT as in the convolution network described earlier. As observed in [2], symmetry of the *cas* function requires the fabrication of only $n/4$ distinct values of conductance in order to implement the matrix D. An approach similar to the one described earlier may be used to fabricate the resistive array in a quantized manner. Once the DHT has been obtained, the Cosine Transform may be derived from this by using an array of summing amplifiers (see equation (3.9)). This approach results in an implementation with minimal time delay involved in computing the transform.

6.4 Analog VLSI Circuit Implementation Of The Classifier Network

The network which we consider here for the final classification of the input signal is one which conforms to the standard model for a pattern classifier feedforward neural network

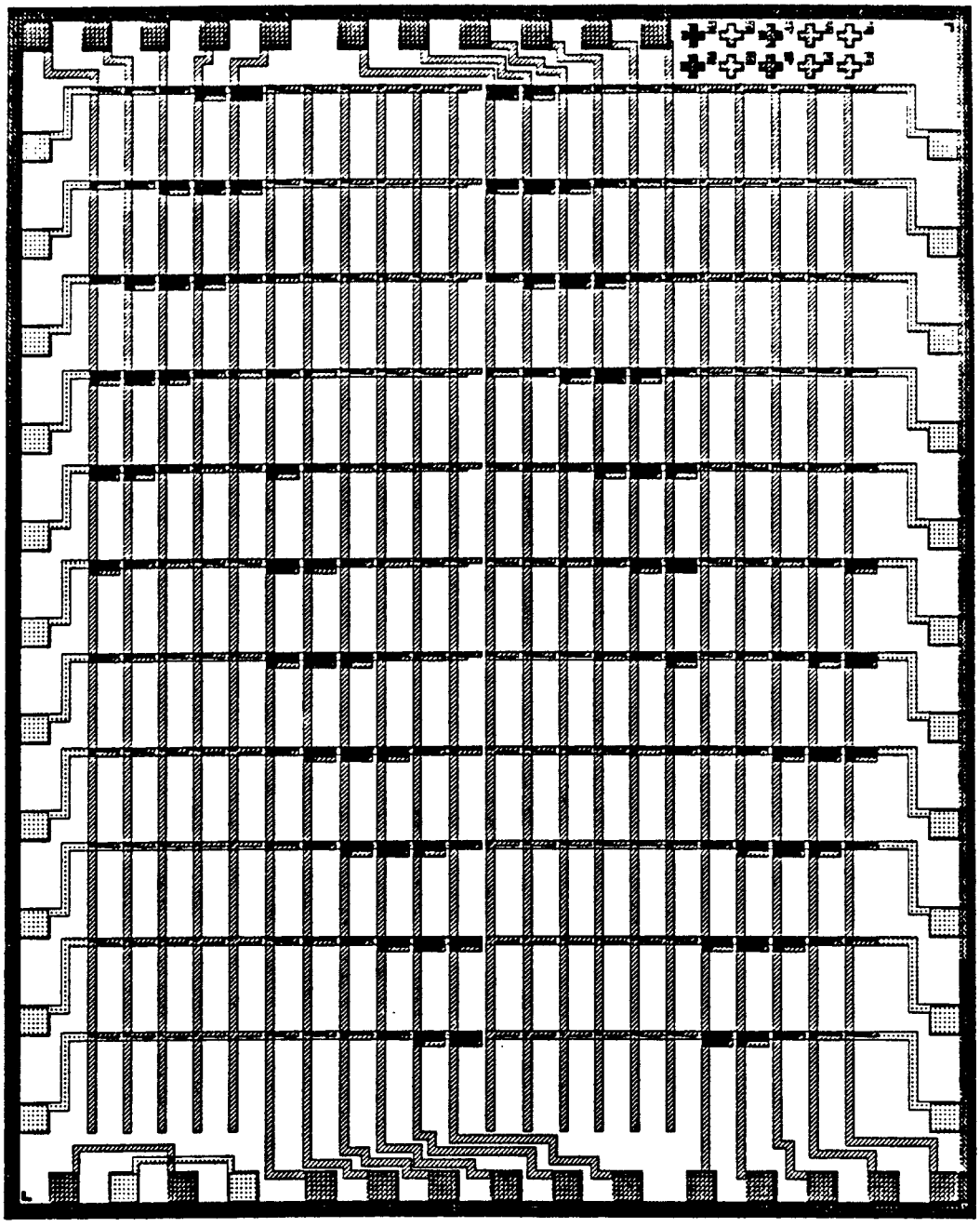


Figure 18: *Layout of resistive interconnection matrix chip.*

(see [4]). Analog VLSI implementation of such layered, feedforward neural networks is a topic of interest to great many researchers in the field and a number of successful implementations of varying complexity have been designed and fabricated (see e.g. [12], [13], [14], and [15]).

Essential to the implementation of an adaptive feedforward neural network is the design and fabrication of *programmable synapses.* We use the term *programmable synapse* to describe a connection between two nodes in which the connection weight can be varied. In terms of analog circuitry, a programmable synapse may be described as a transconductance amplifier with variable gain. It is also essential that circuit complexity of a programmable synapse be minimal. In [16] a hybrid digital-analog array of programmable synapses is described which has been fabricated and is currently being tested at the Naval Research Laboratories in Washington D.C.. As will be discussed later, the particular digital-analog design in [16] is amenable to the overall network configuration which we consider.

One major obstacle in VLSI implementation of a feedforward neural network is that as the network grows in size (number of nodes and layers), full interlayer connectivity [4] is intractable in terms of both allocation of silicon real estate and routing of the wires required to implement connections. In this section we discuss a VLSI design strategy and architecture which permits a great deal of flexibility with respect to the number of layers in the network while maintaining full connectivity and programmability of the connections. We call the architecture which we describe here *STAACNNET* for *STackable Adaptive Analog Circuit Neural NETwork* .

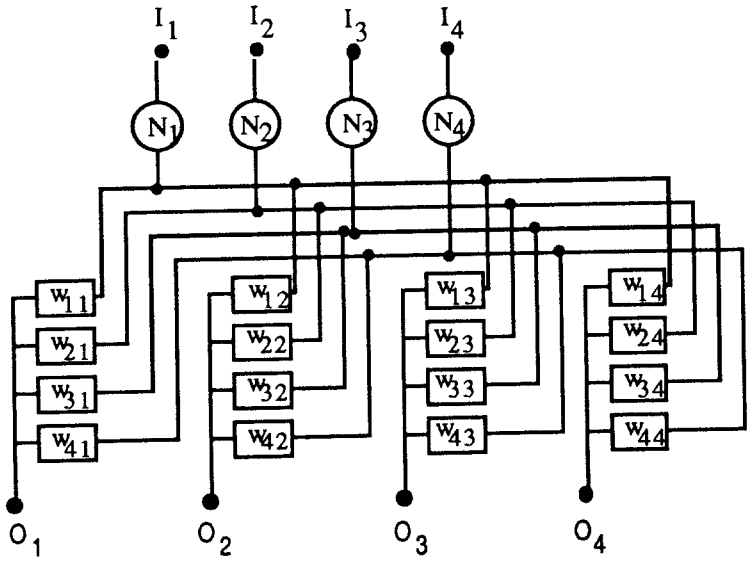
6.4.1 STAACNNET

Among the more appealing attributes of neural network architectures is that the networks are composed of a large number of essentially identical elements. In the case of the classifier network which we consider here, the network is composed of a number of identical layers[5]. Each layer is composed of a number of identical nodes followed by a complete set of connections to the next layer of nodes. In VLSI implementation of multilayer, feedforward neural networks, this uniformity may be exploited to allow implementation of large networks as a cascade of less complex elements.

For implementation of the classifier network , we propose an architecture of the form depicted in Figure 19. In this architecture, a single layer of nodes, together with a complete set of connections to the next layer, is implemented on a single chip. Hence if K_N is the number of nodes in a layer, a single chip would incorporate K_N nodes (with inputs provided to each of these nodes), K_N^2 programmable connections, and K_N outputs to be connected to the inputs of the next layer. Each chip would be packaged individually and fitted with a connector which connects to the pins on the package and also provides a socket into which can be placed a second such chip. Since the connections weights are programmable it is necessary to be able to select a particular connection whose weight is to be modified. If K_l is the maximum number of layers we are to implement in a single

[4]Full interlayer connectivity in this case refers to the fact that every node in a layer is connected to every node in the layer preceding and the layer following it.

[5]A layer with a smaller number of nodes may be implemented by simply setting appropriate connections to zero.



Single layer of network with programmable synapses (4 node)

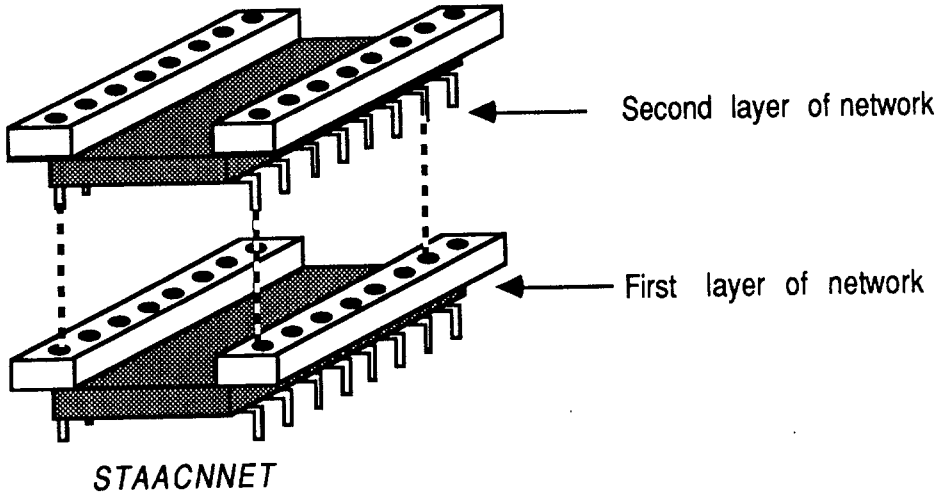


Figure 19: *Implementation architecture for classifier network*

network, we require $\log_2 K_l$ address lines to select a layer, and $2\log_2(K_N)$ address lines to select a connection within a selected layer. Hence the total number of pins required on a package is[6]

$$2K_N + \log_2(K_N^2) + \log_2(K_l) + 4.$$

For example if we would like to use 20 nodes in a layer and allow up to four layers in the network, 55 pins are required per chip. Note that the physical stacking of the chips is just a convenient feature and not a necessity of this architecture, since an equivalent configuration may be obtained by simply externally connecting the appropriate pins.

Training algorithms such as the backpropagation algorithm would be implemented in a microprocessor environment from which individual connection weights could be selected and updated.

The advantages of this structure are the following: (i)reduced density requirements for individual chips, (ii)experimentation with number of layers and number of nodes in different layers is facilitated, and (iii)full programmability of the connection patterns and flexibility with learning rules since training algorithms are microprocessor controlled.

7 Discussion and Conclusions

We have proposed an analog neural network model of a real-time system for the decomposition of superposed radar returns in the presence of noise. This system consists of three distinctly identifiable processes which correspond to the operations of noise reduction, data transformation, and classification respectively. Noise reduction via a non-linear neural network model exhibits highly robust performance against noise level (signal to noise ratio) in the input. It has further been shown that the neural network nonlinear noise filtering is far superior to a simple low pass filter implementation.

Preliminary simulations favorably indicate the success of the proposed architecture. Furthermore, the invocation of existing optimization based design tools promises to improve the performance of such a system. Most applications of neural networks for classification tasks have been previously restricted to binary decision regions (simple yes/no answers to an arbitrary number of hypotheses). Here, we have demonstrated a neural architecture capable of performing classification on a *continuous* decision space. Such classification has been demonstrated to perform to a high degree of accuracy.

Some of the most significant benefits of the proposed system are a result of implementation of the system as an analog VLSI circuit. Foremost among these benefits is processing speed. A breadboard prototype of the MEDN [7] has demonstrated the networks capability to converge to solutions in under 10μsec. VLSI implementation of the MEDN is underway at the Naval Research Lab in Washington D.C.

[6]The additional 4 pins are required for power supply, ground, and programming voltage for the connection weights.

References

[1] R. P. Gorman and T. J. Sejnowski, "Learned classification of sonar targets using a massively parallel network," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, July 1988.

[2] A. D. Culhane, M. C. Peckerar, and C. R. K. Marrian, "A neural net approach to discrete fourier transforms," Preprint. Naval Research Laboratory, Code 6804, Washington D.C., 1987.

[3] R. Bracewell, *The Hartley Transform.* Oxford University Press, 1986.

[4] D. Rumelhart and G. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Cambridge, Ma.: MIT Press, 1988.

[5] S. C. Ahalt, F. D. Garber, I. Jouny, and A. K. Krishnamurthy, "Performance of synthetic neural network classification of noisy radar signals," IEEE NIPS Poster, 1988.

[6] C. Marrian and M. Peckerar, "Electronic neural net algorithm for maximum entropy deconvolution," in *Proceedings IEEE First Annual Inter. Conf. On Neural Networks, San Diego, Ca.*, June 1987.

[7] Y. C. Pati, D. Friedman, P. S. Krishnaprasad, C. T. Yao, M. Peckerar, R. Yang, and C. R. K. Marrian, "Neural networks for tactile perception," in *Proceedings IEEE Inter. Conf. On Robotics and Automation, Philadelphia, Pa.*, pp. 134–139, April 1988.

[8] T. Poggio and C. Koch, "Ill-posed problems in early vision: from computational theory to analog networks," in *Proceedings Royal Society of London*, pp. 303–323, 1985.

[9] M. Fan, J. Koninckx, L. Wang, and A. Tits, "Console: a cad tandem for optimization-based design interacting with arbitrary simulators," Technical Report, University of Maryland, Systems Research Center, 1987.

[10] C. Marcus and R. Westervelt, "Stability of analog neural networks with delay," In Press. Physical Review A.

[11] C. R. K. Marrian, M. C. Peckerar, I. Mack, and Y. C. Pati, "Electronic neural net for solving ill-posed problems with anentropy regularizer," in *Proceedings 8th Inter. Maximum Entropy Workshop*, (Cambridge, U.K.), Kluwer, 1988. J. Skilling Editor.

[12] Y. S. Abu-Mostafa and D. Psaltis, "Optical neural computers," *Scientific American*, pp. 88–95, March 1987.

[13] H. P. Graf, L. D. Jackel, and W. E. Hubbard, "Vlsi implementation of a neural network model," *IEEE Computer Magazine*, pp. 41–49, March 1988.

[14] L. Jackel, R. Howard, H. Graf, B. Straughn, and J. Denker, "Artificial neural networks for computing," *J. Vac. Sci. Technol. B*, vol. 4, pp. 61–63, January/February 1986.

[15] D. Schwartz, R. Howard, J. Denker, R. Epworth, H. Graf, W. Hubbard, L. Jackel, B. Straughn, and D. Tennant, "Dynamics of microfabricated electronic neural networks," *Appl. Phys. Lett*, vol. 50(16,20), April 1987.

[16] F. Kub, I. Mack, K. Moon, C. Yao, and J. Modolo, "Programmable analog synapses for microelectronic neural networks using a hybrid digital-analog approach," in *Proceedings IEEE International Conference On Neural Networks, San Diego, Ca.*, July 1988.

[17] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci., U.S.A.*, vol. 79, pp. 2554–2558, April 1982.

[18] D. Tank and J. Hopfield, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

[19] M. W. Hirsch, "Convergence in neural nets," in *Proceedings IEEE International Conference On Neural Networks, San Diego, Ca.*, June 1987.

[20] J. Hopfield, "Neurons with graded responses have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. Sci., U.S.A.*, vol. 81, pp. 3088–3092, May 1984.

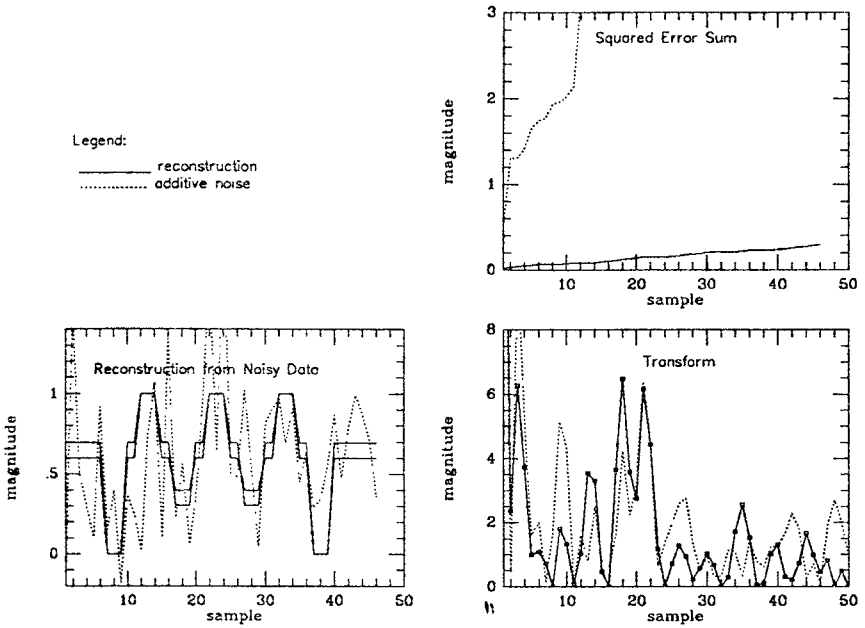
[21] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, pp. 4–22, April 1987.

[22] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, no. 6035, pp. 314–319, 0000.

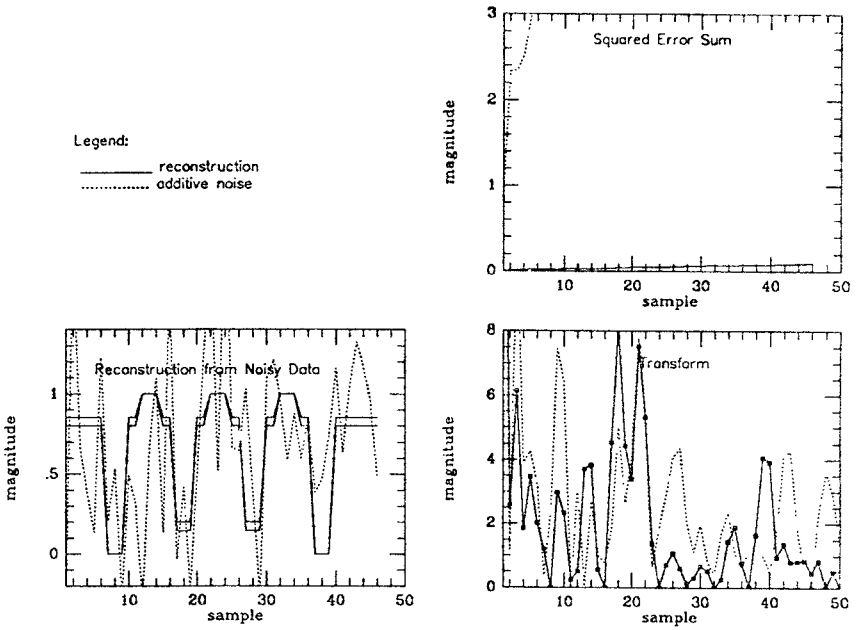
APPENDIX A: *Simulation Runs*

S/N = 0.0 dB

Signal: p46

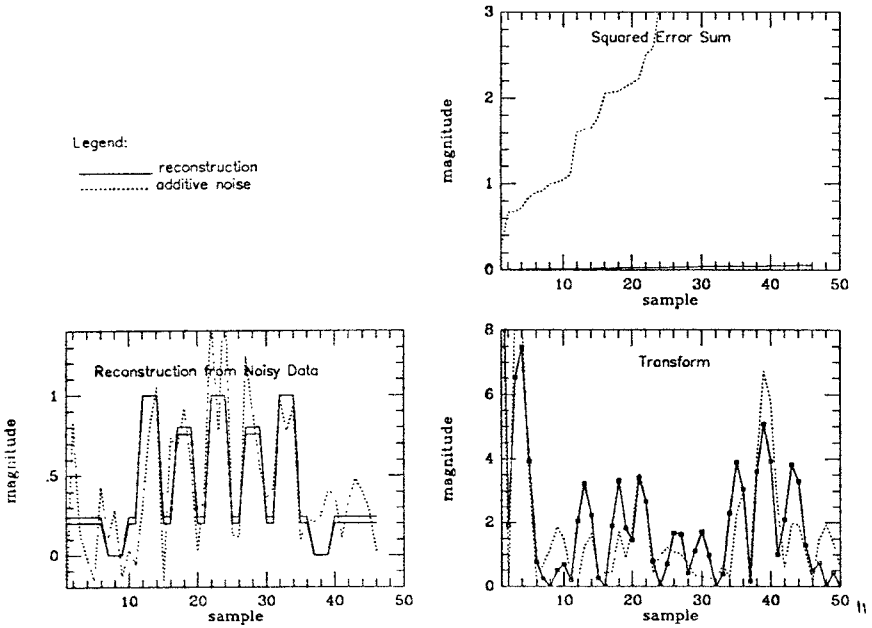


Signal: p28

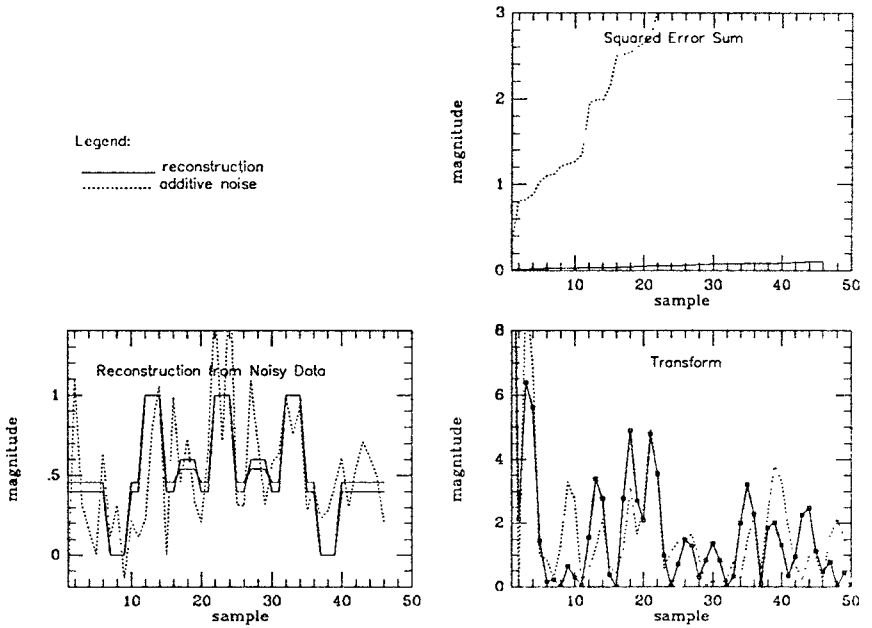


S/N = 0.0 dB

Signal: p82

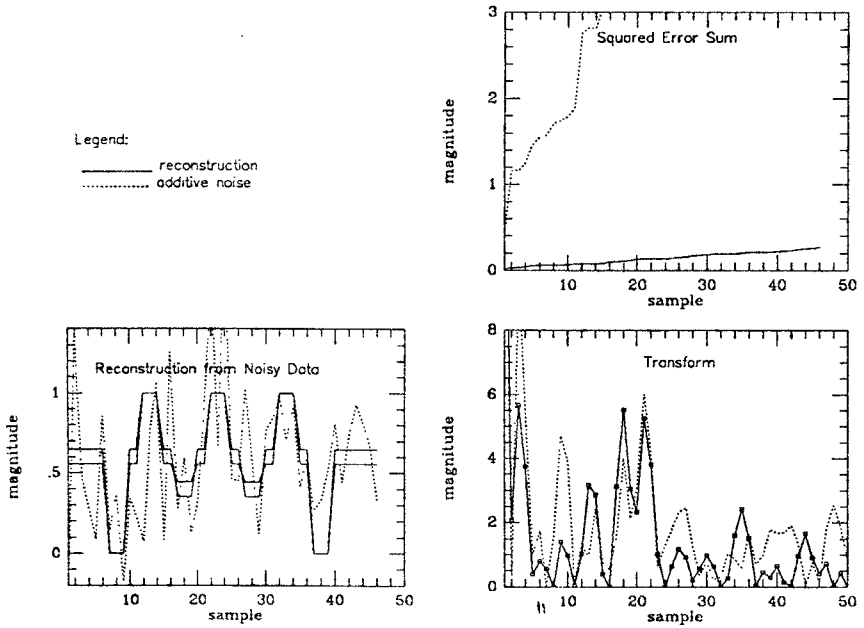


Signal: p64

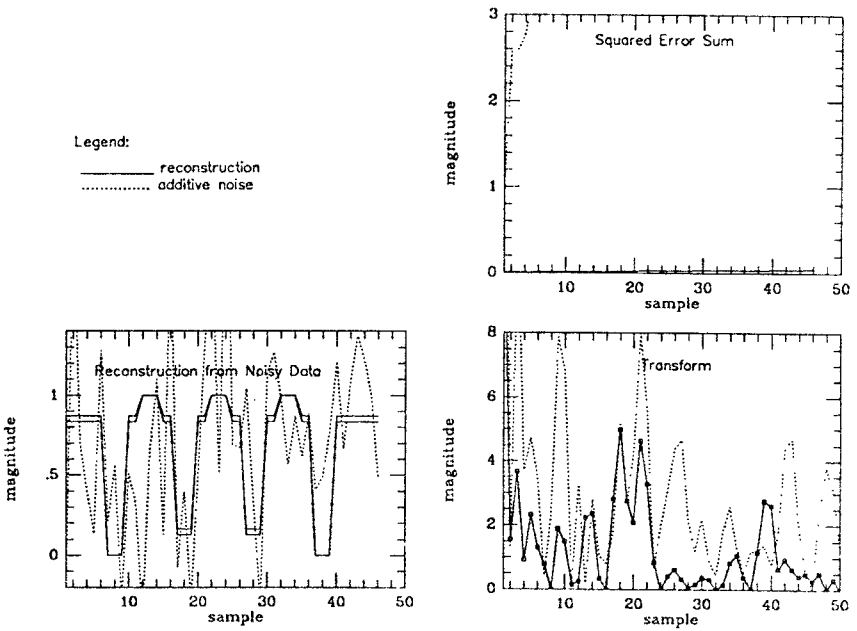


S/N = 0.0 dB

Signal: p45

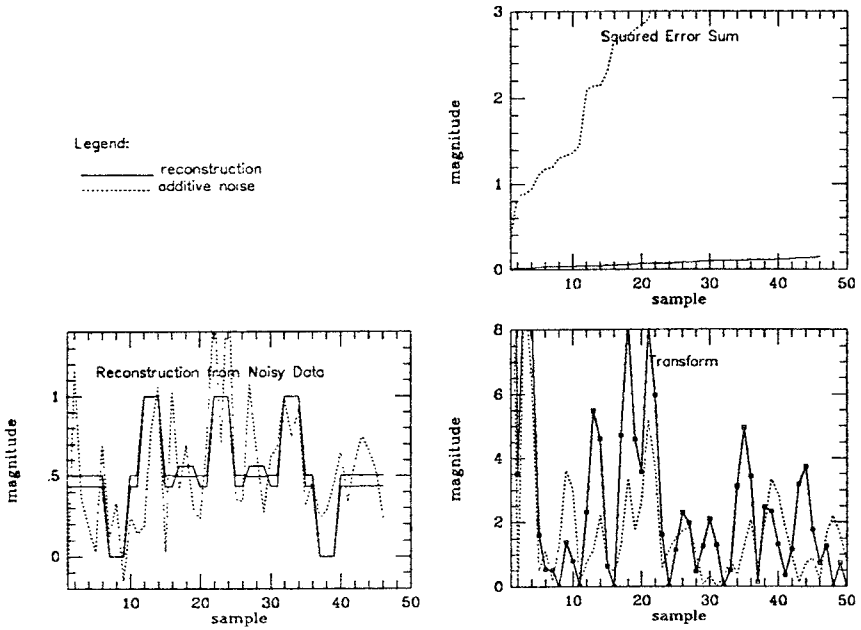


Signal: p15

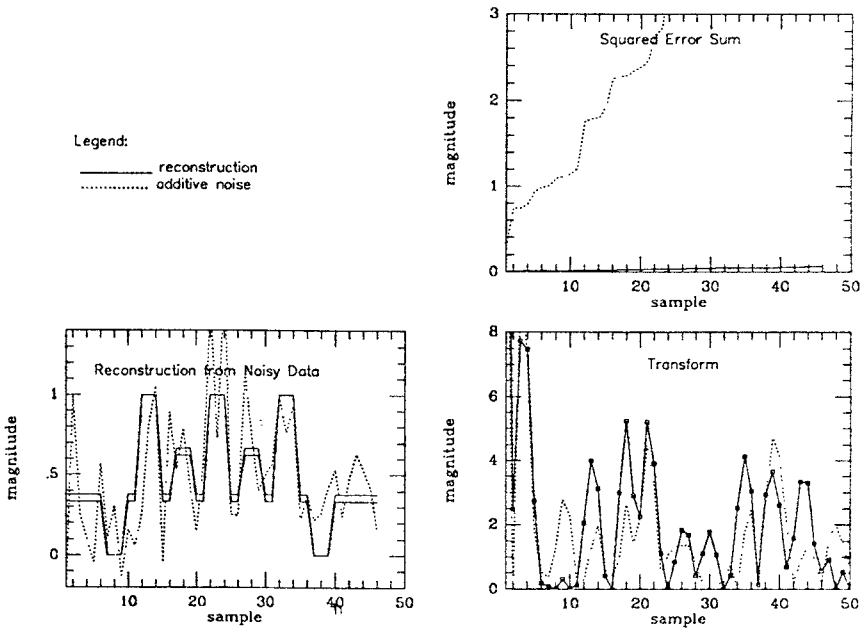


S/N = 0.0 dB

Signal: p97

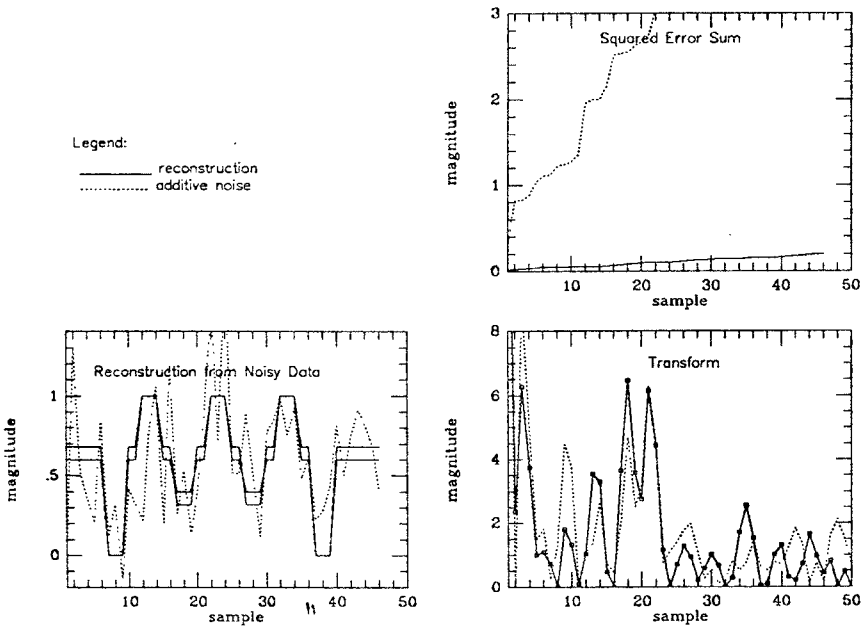


Signal: p84

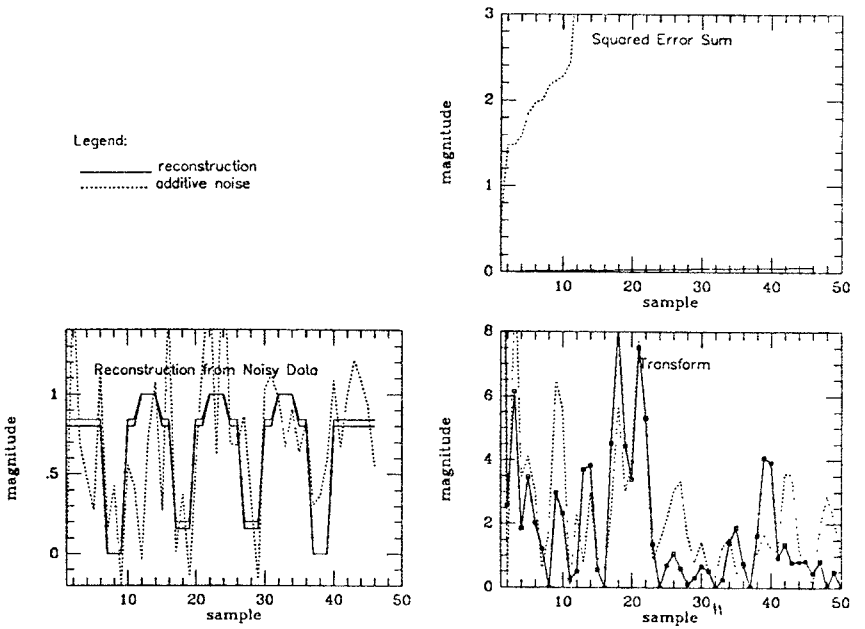


S/N = 1.0 dB

Signal: p46

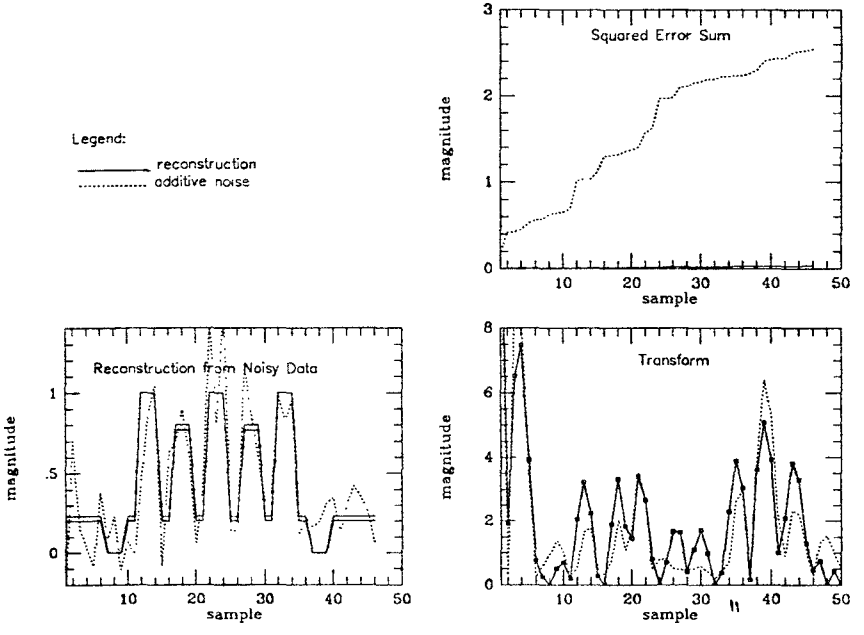


Signal: p28

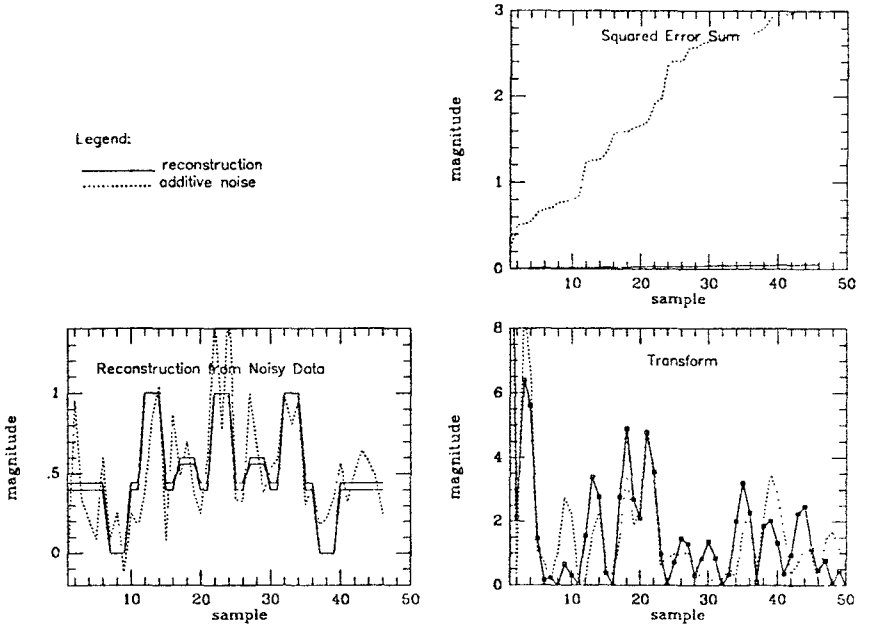


S/N = 1.0 dB

Signal: p82

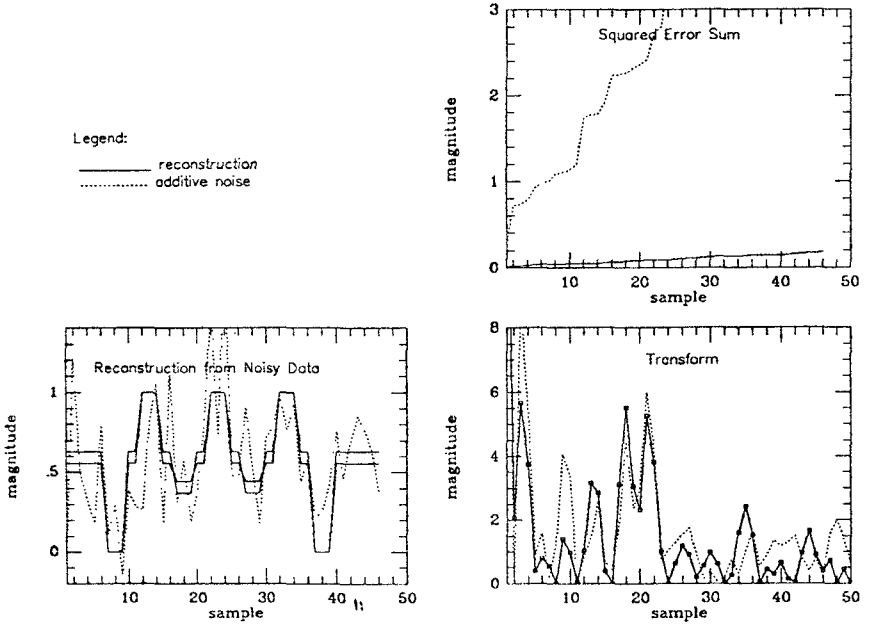


Signal: p64

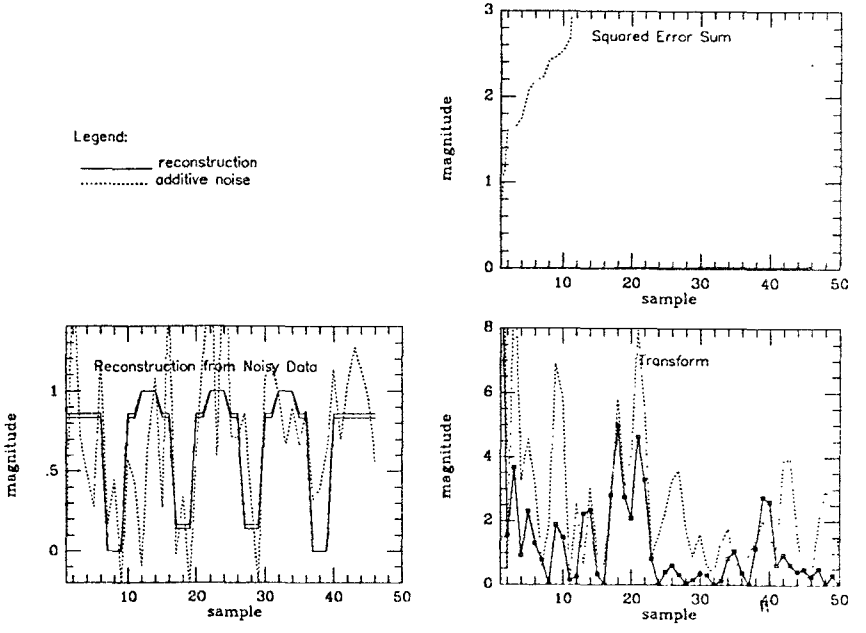


S/N = 1.0 dB

Signal: p45

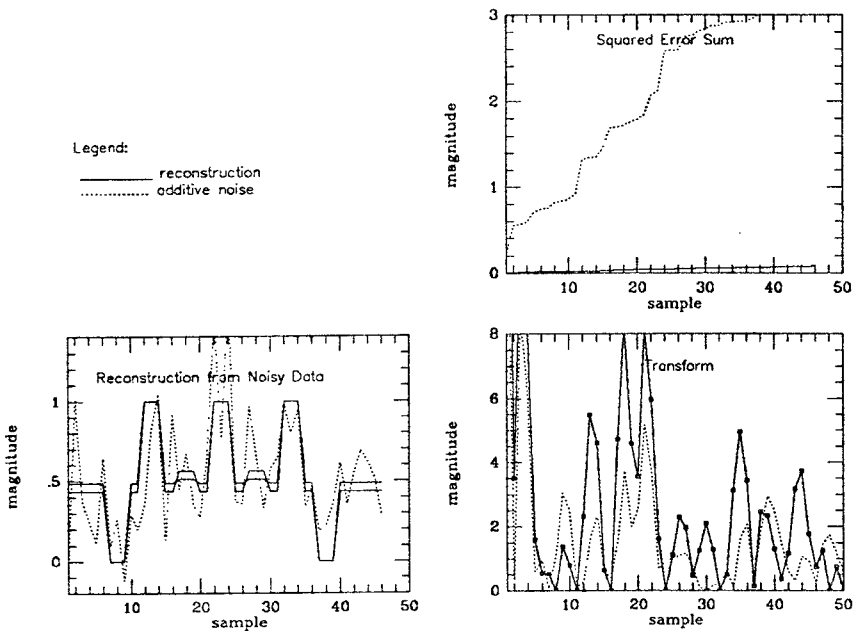


Signal: p15

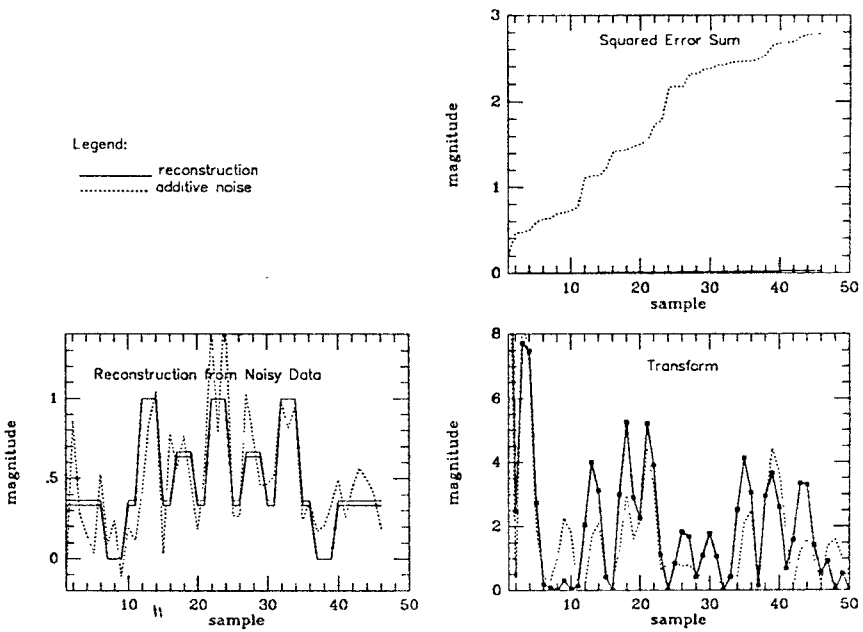


S/N = 1.0 dB

Signal: p97

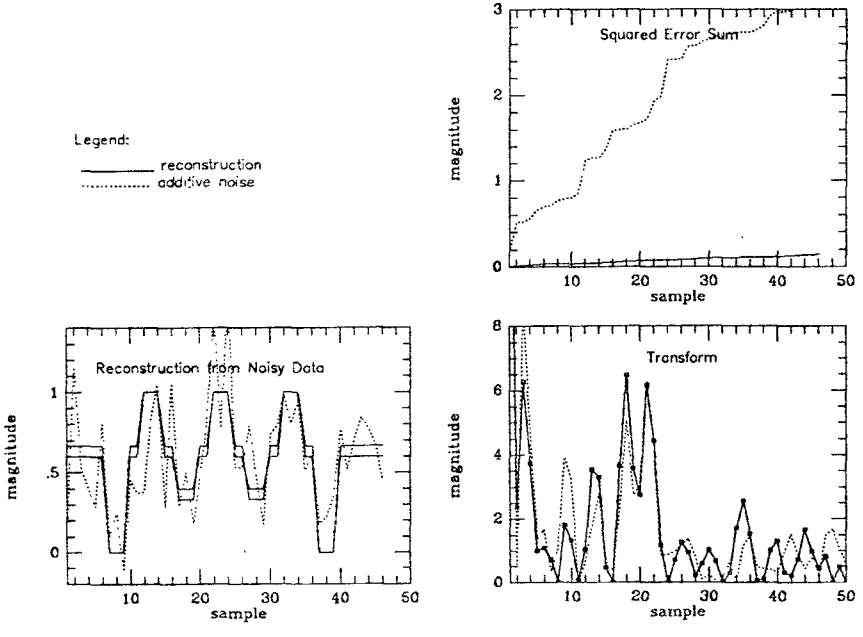


Signal: p84

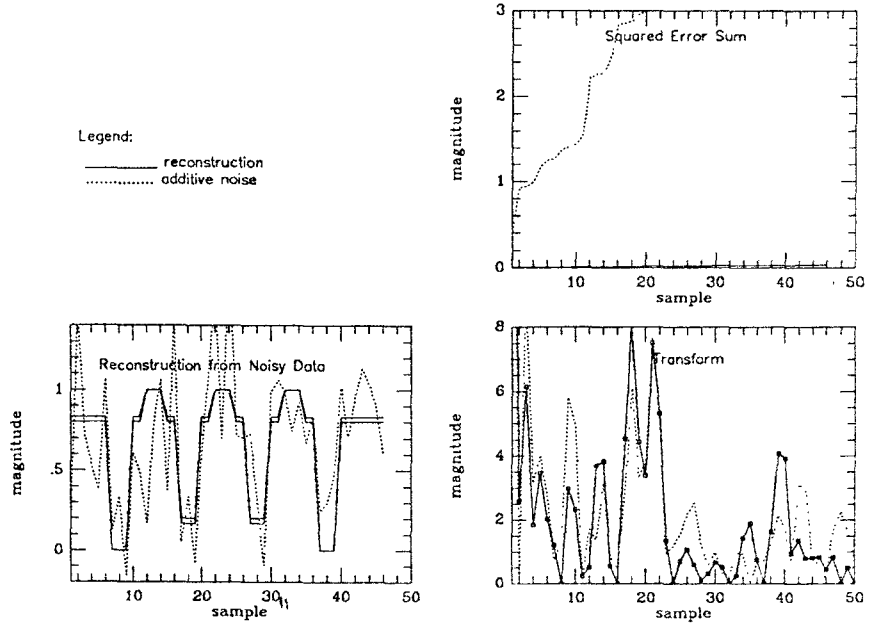


S/N = 2.0 dB

Signal: p46

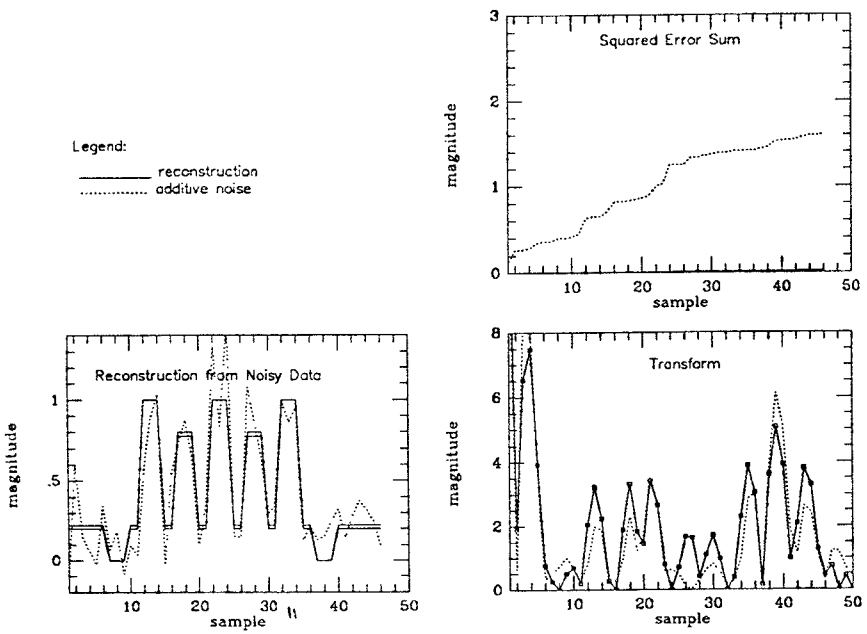


Signal: p28

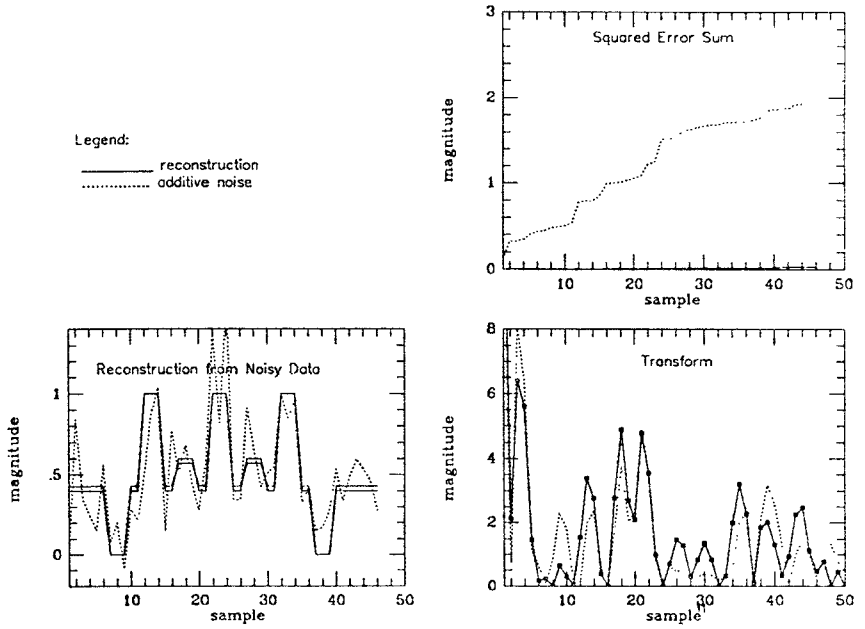


S/N = 2.0 dB

Signal: p82

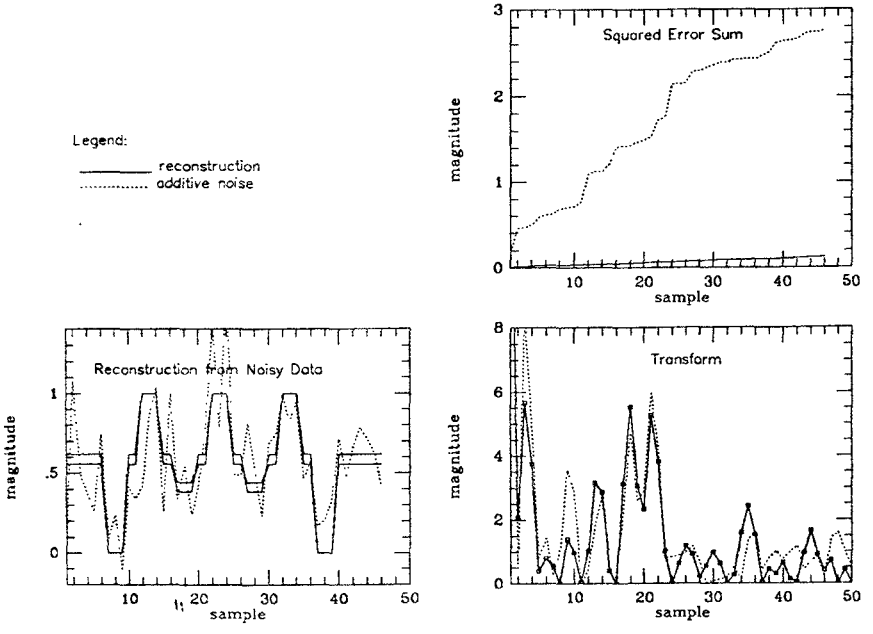


Signal: p64

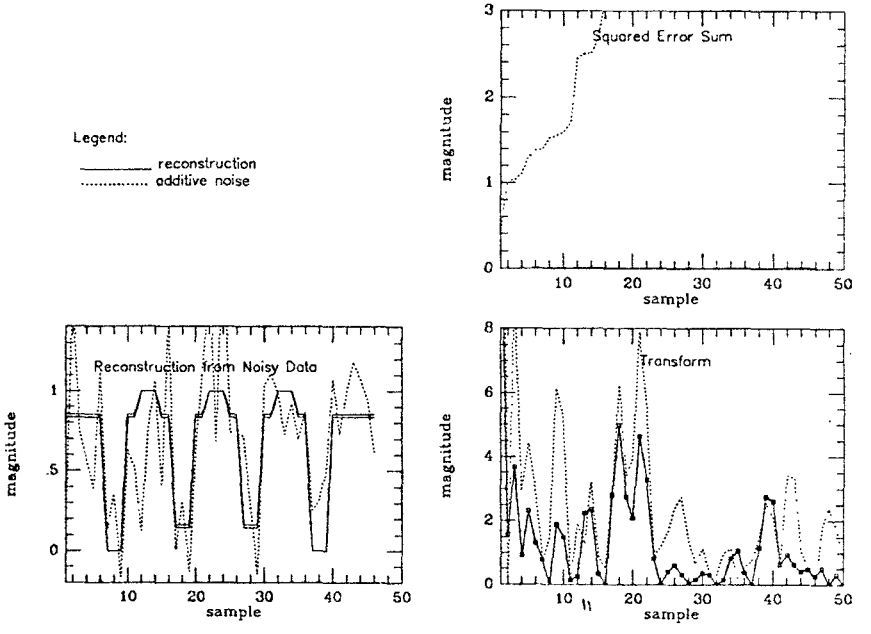


S/N = 2.0 dB

Signal: p45

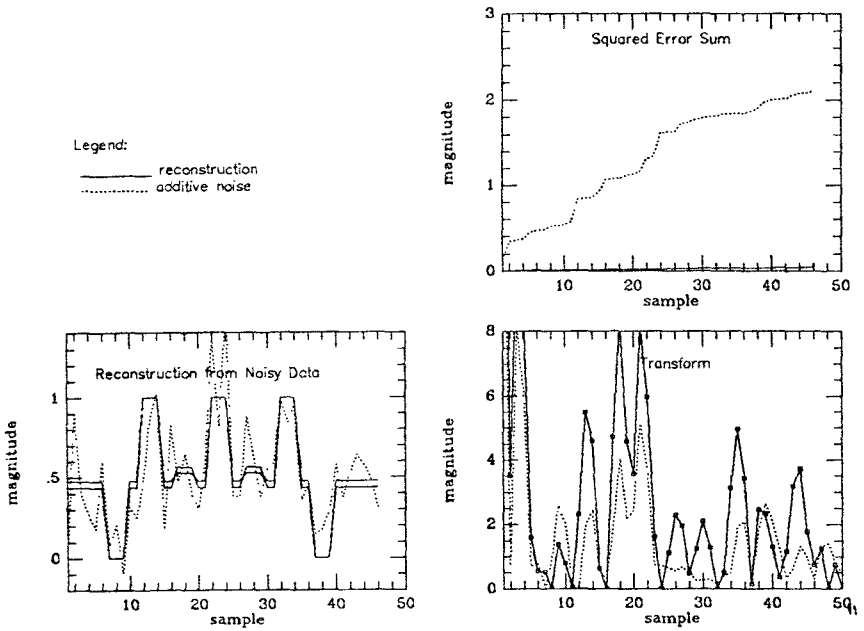


Signal: p15

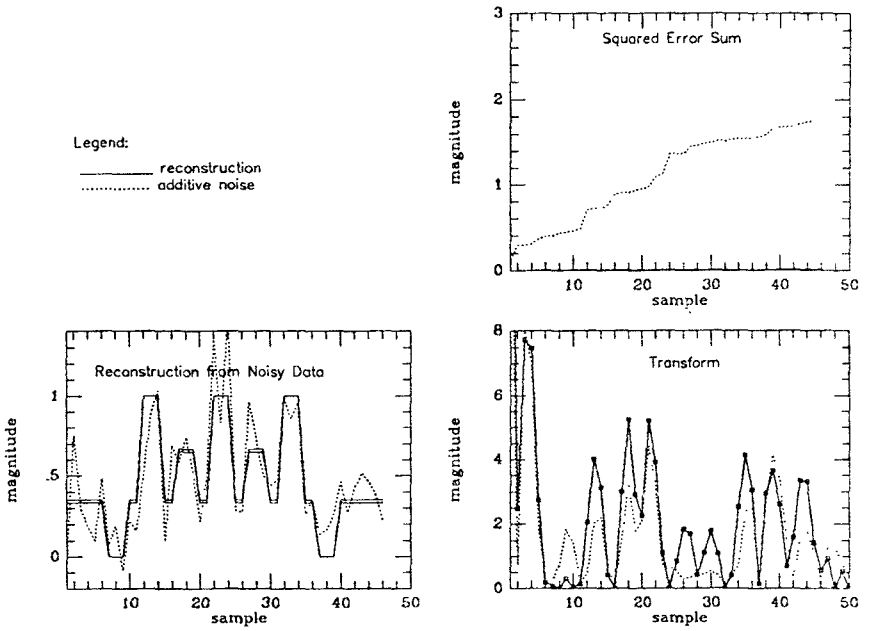


S/N = 2.0 dB

Signal: p97

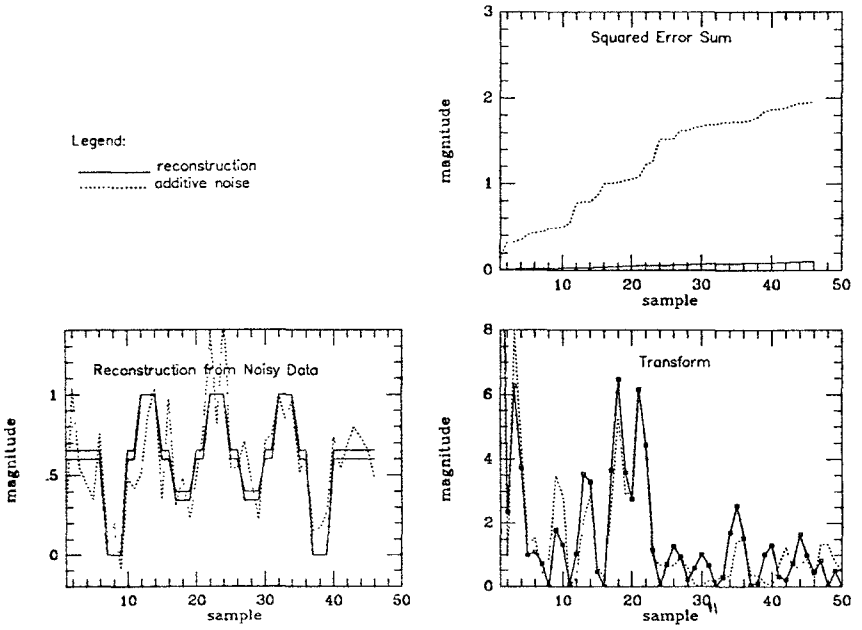


Signal: p84

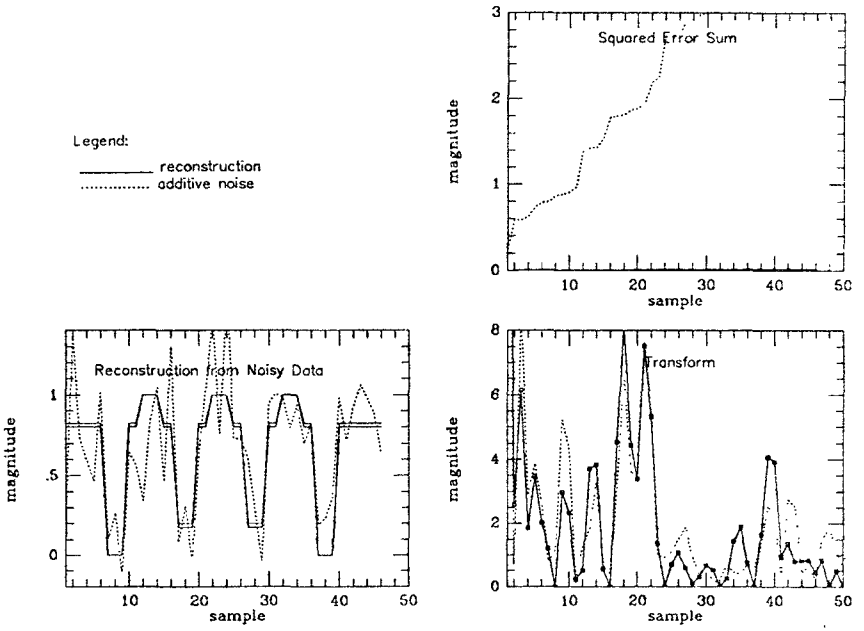


S/N = 3.0 dB

Signal: p46

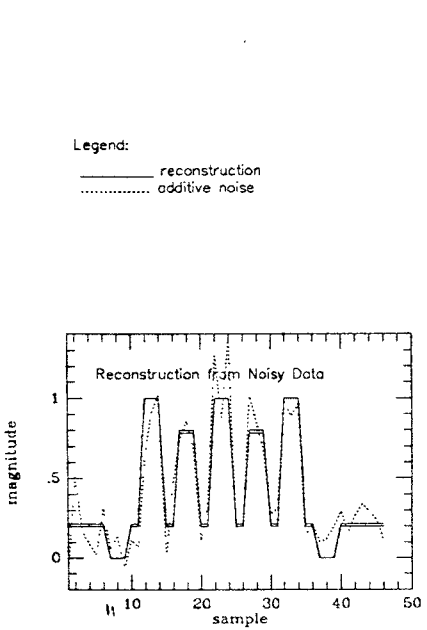


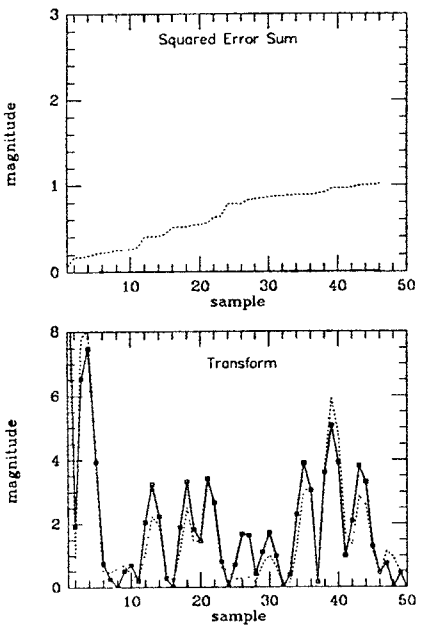
Signal: p28



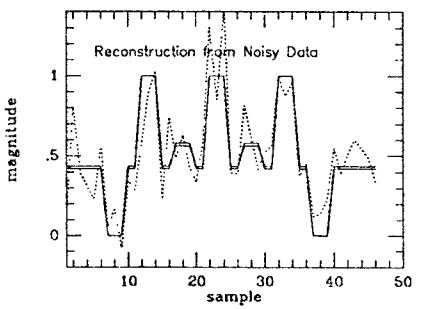
S/N = 3.0 dB

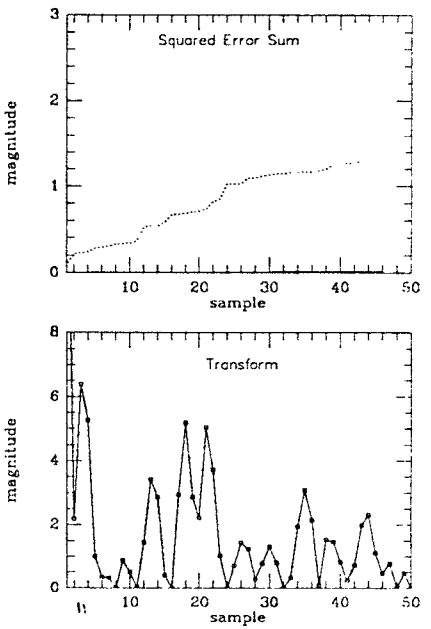
Signal: p82





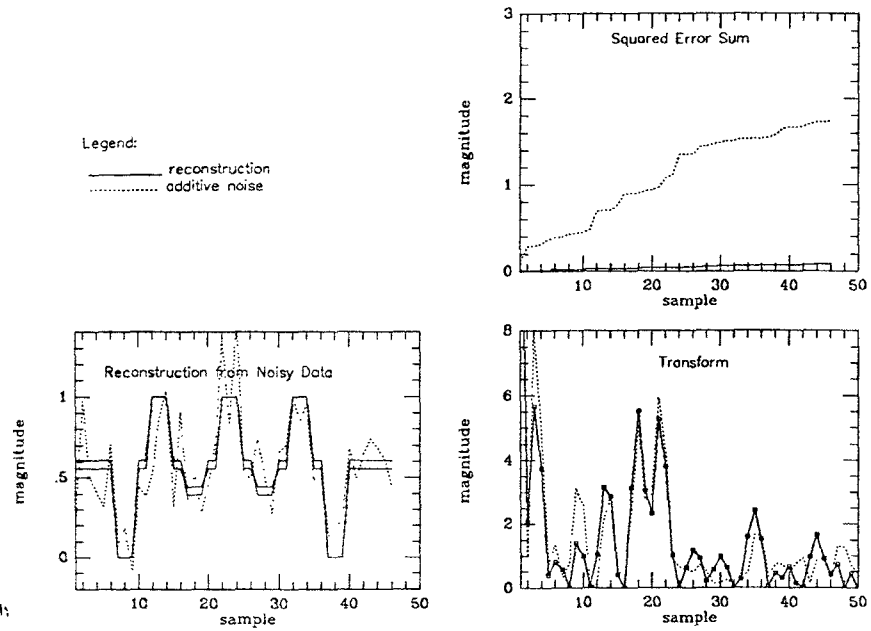
Signal: p64



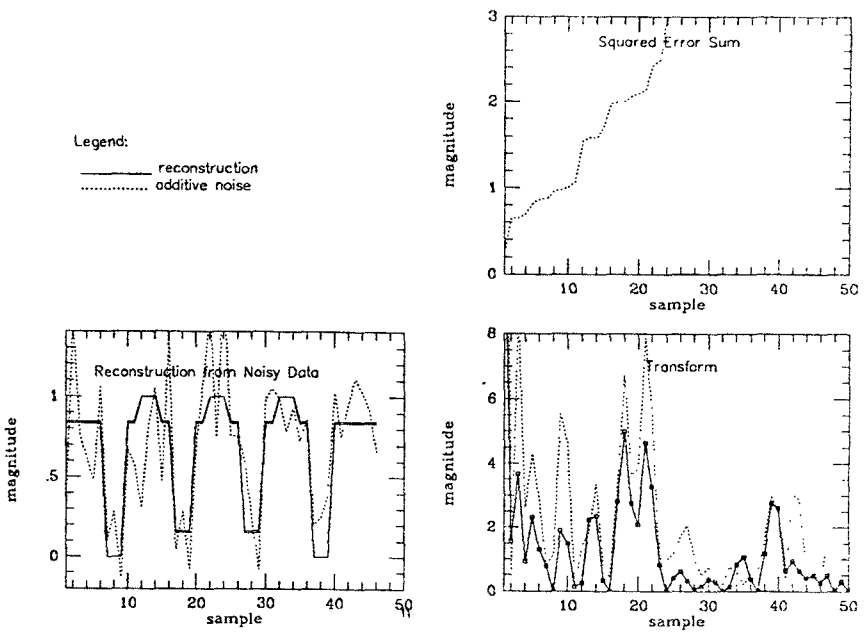


S/N = 3.0 dB

Signal: p45

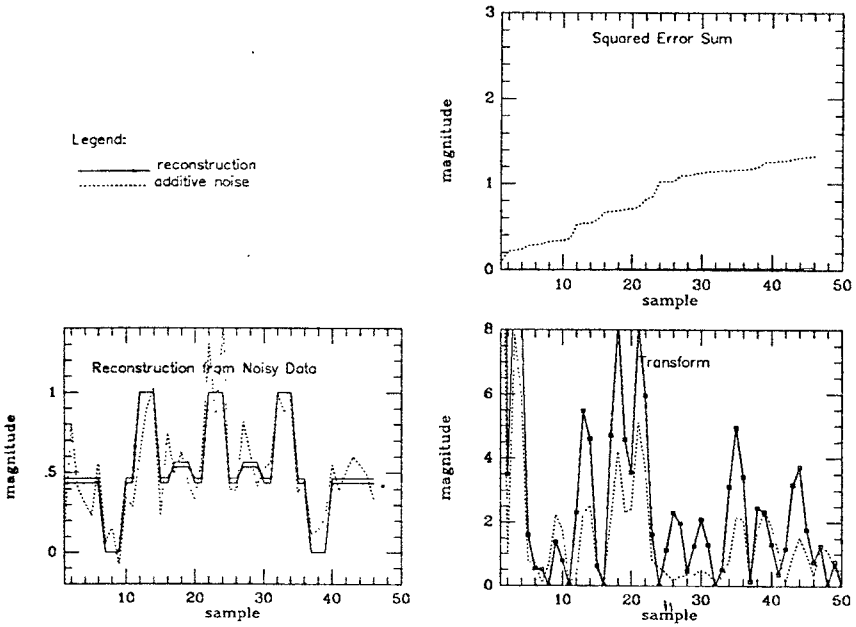


Signal: p15

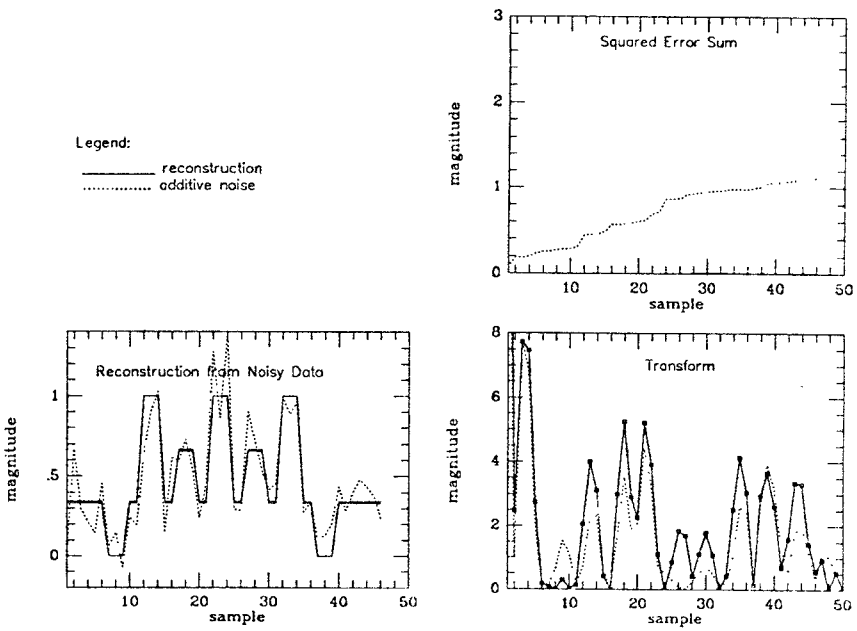


S/N = 3.0 dB

Signal: p97

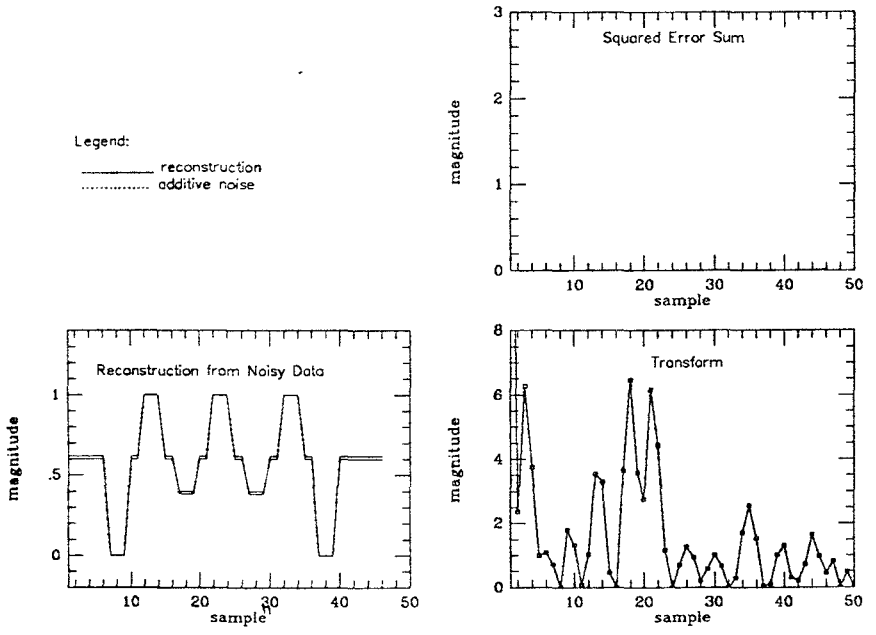


Signal: p84

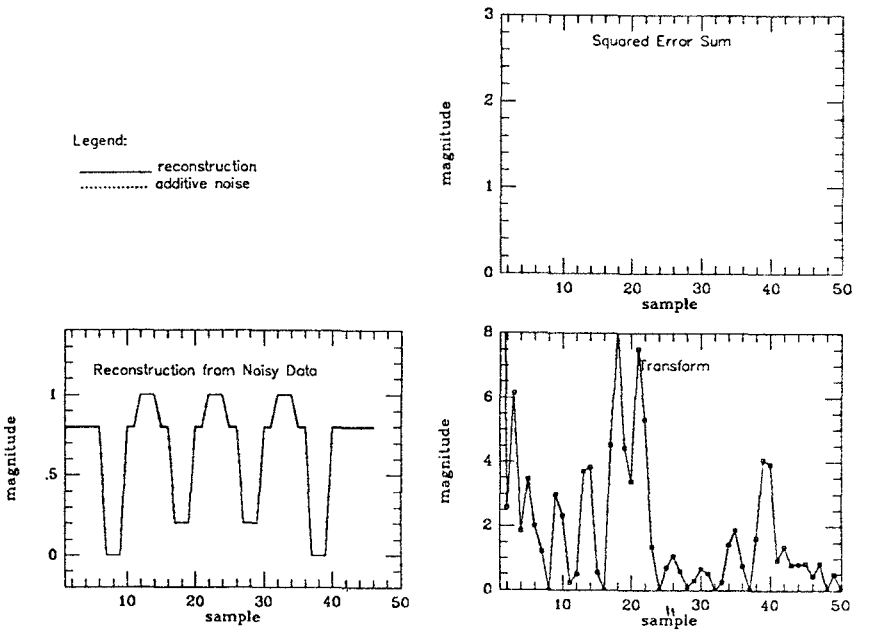


$S/N = \infty$ dB

Signal: p46

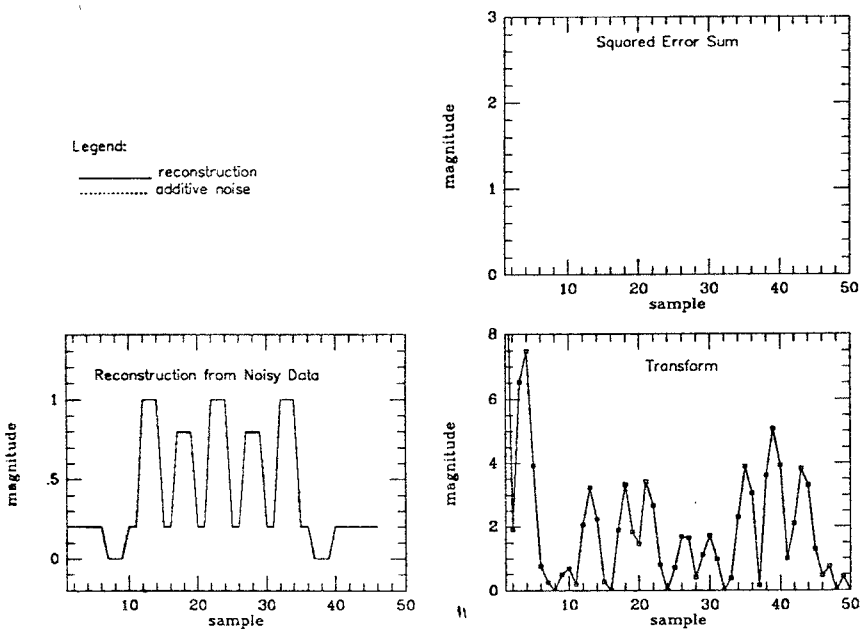


Signal: p28

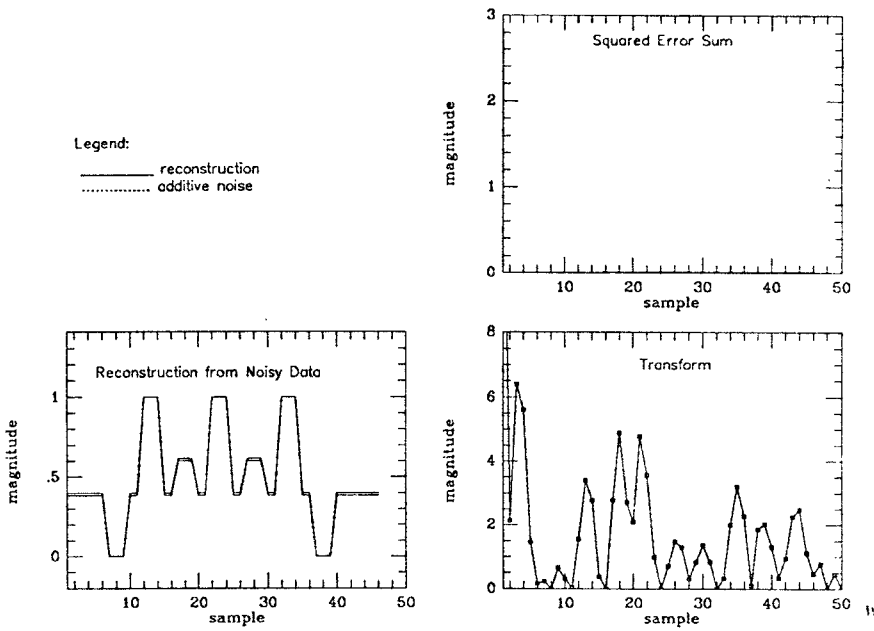


$S/N = \infty$ dB

Signal: p82

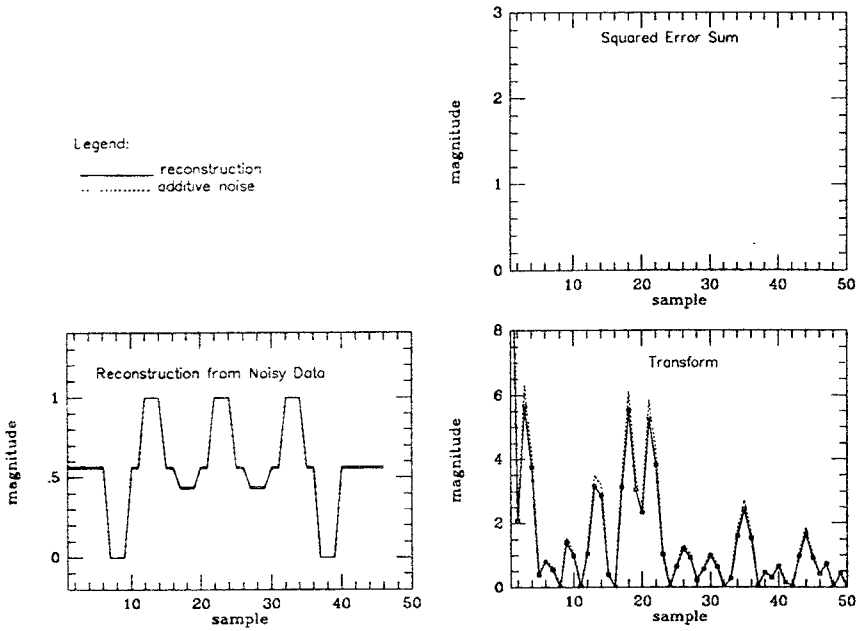


Signal: p64

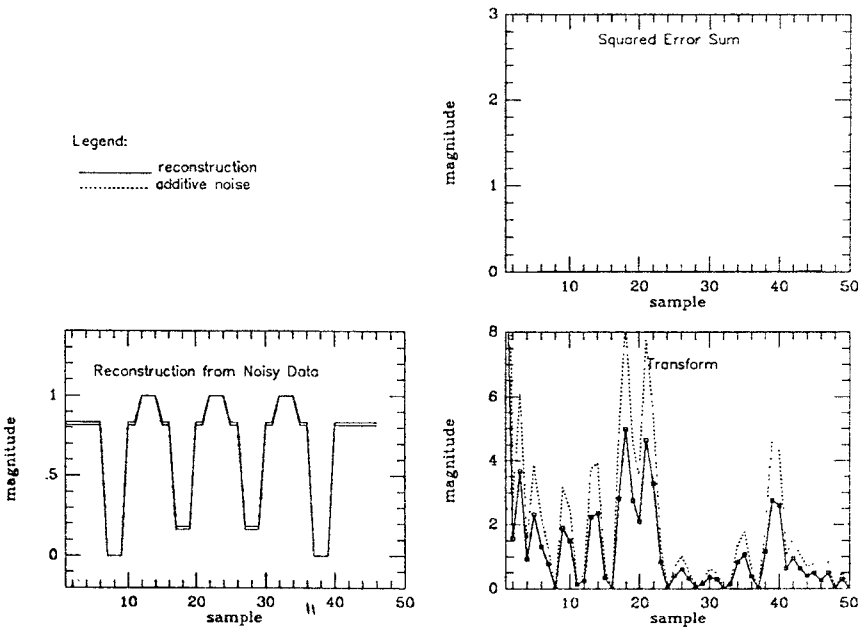


$S/N = \infty$ dB

Signal: p45

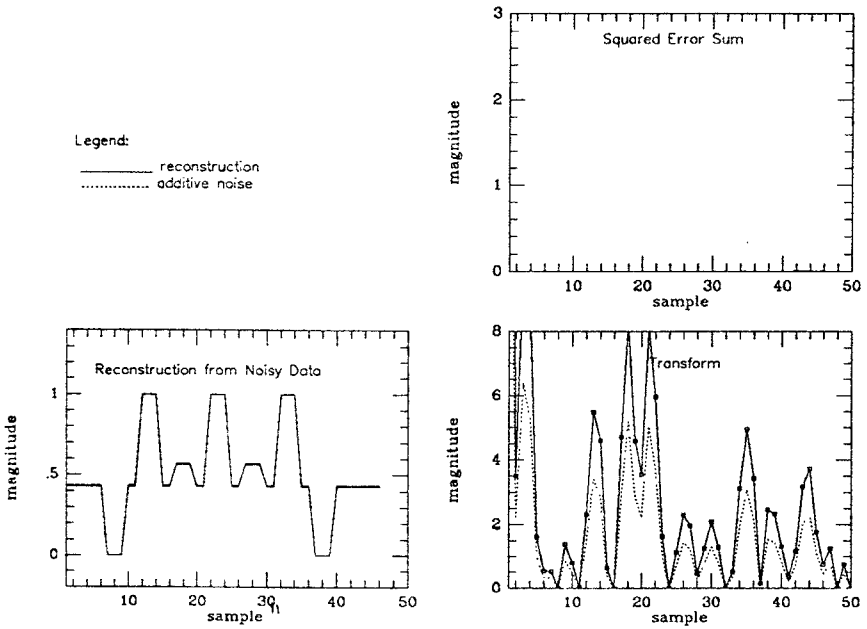


Signal: p15



$S/N = \infty$ dB

Signal: p97



Signal: p84