LECTURES ON

NEURAL NETWORKS

BY

PROF. BERNARD WIDROW

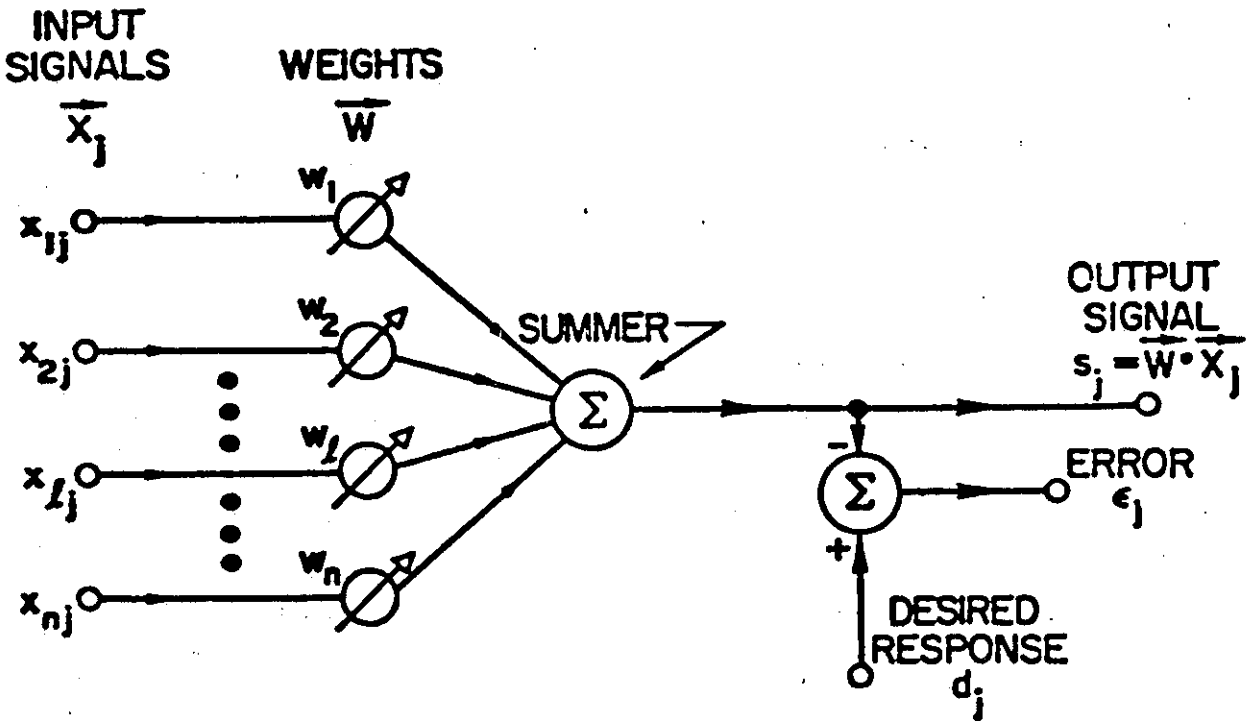
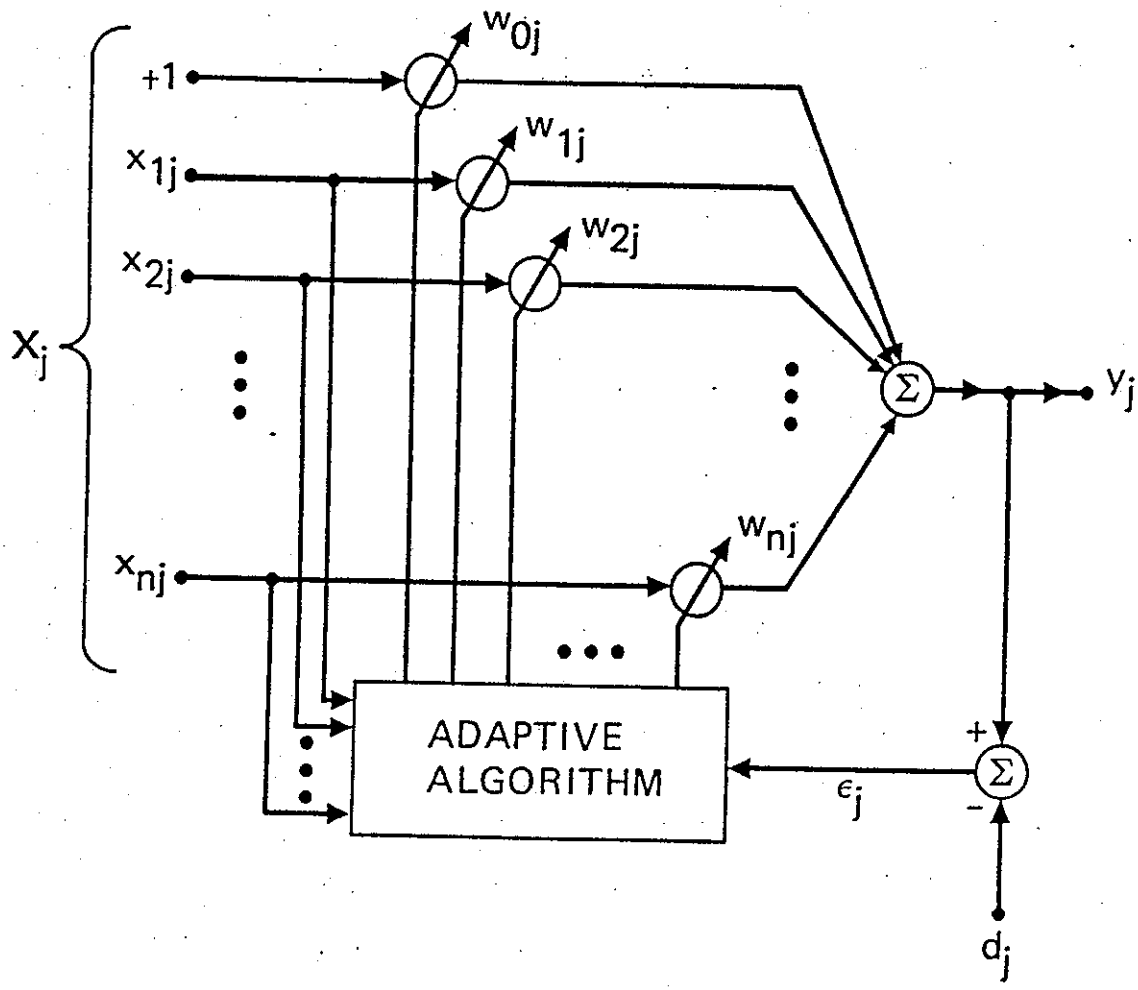
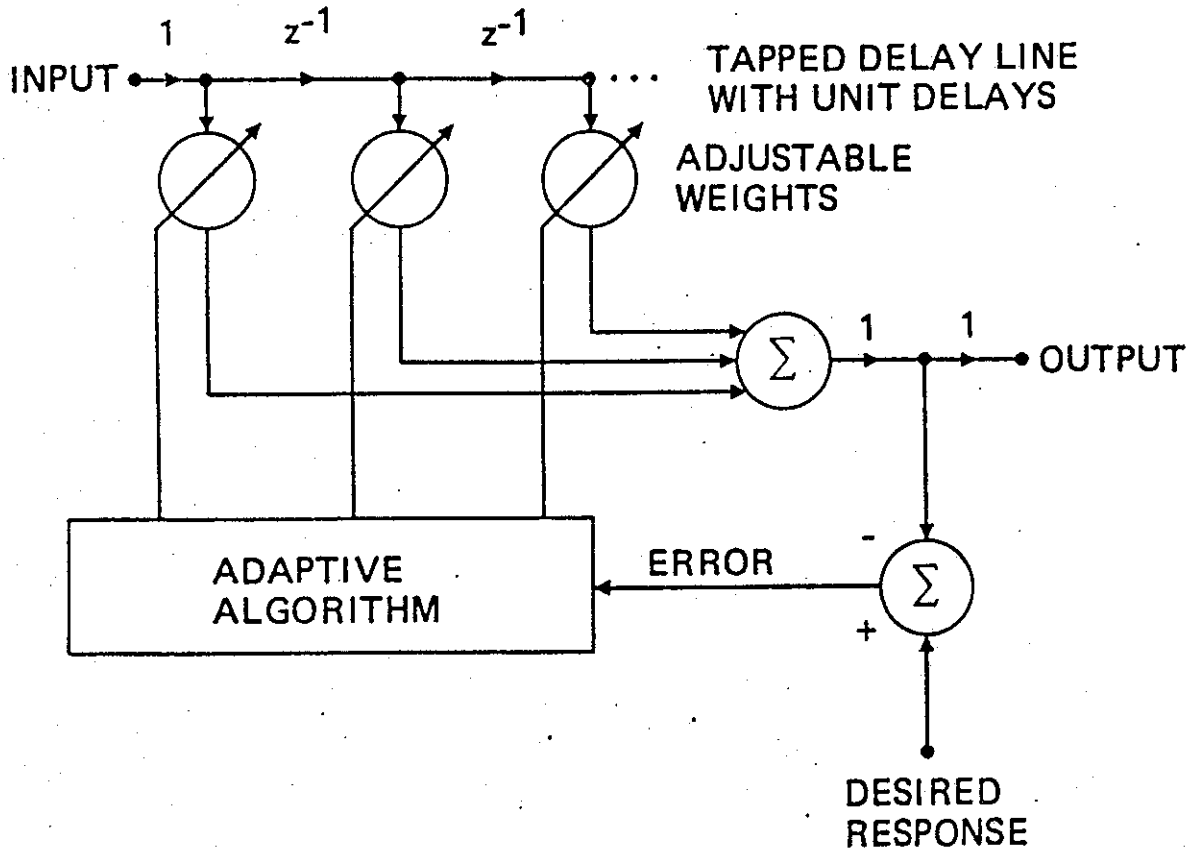


Fig. 2. Adaptive linear combinatorial system.


w_{0j}
+1
w_{1j}
x_{1j}
w_{2j}
x_{2j}
x_j
y_j
w_{nj}
x_{nj}
ADAPTIVE
ALGORITHM
+
ϵ_j
−
d_j


1
z^{-1}
z^{-1}
INPUT
TAPPED DELAY LINE
WITH UNIT DELAYS
ADJUSTABLE
WEIGHTS
1
1
OUTPUT
ADAPTIVE
ALGORITHM
ERROR
-
+
DESIRED
RESPONSE

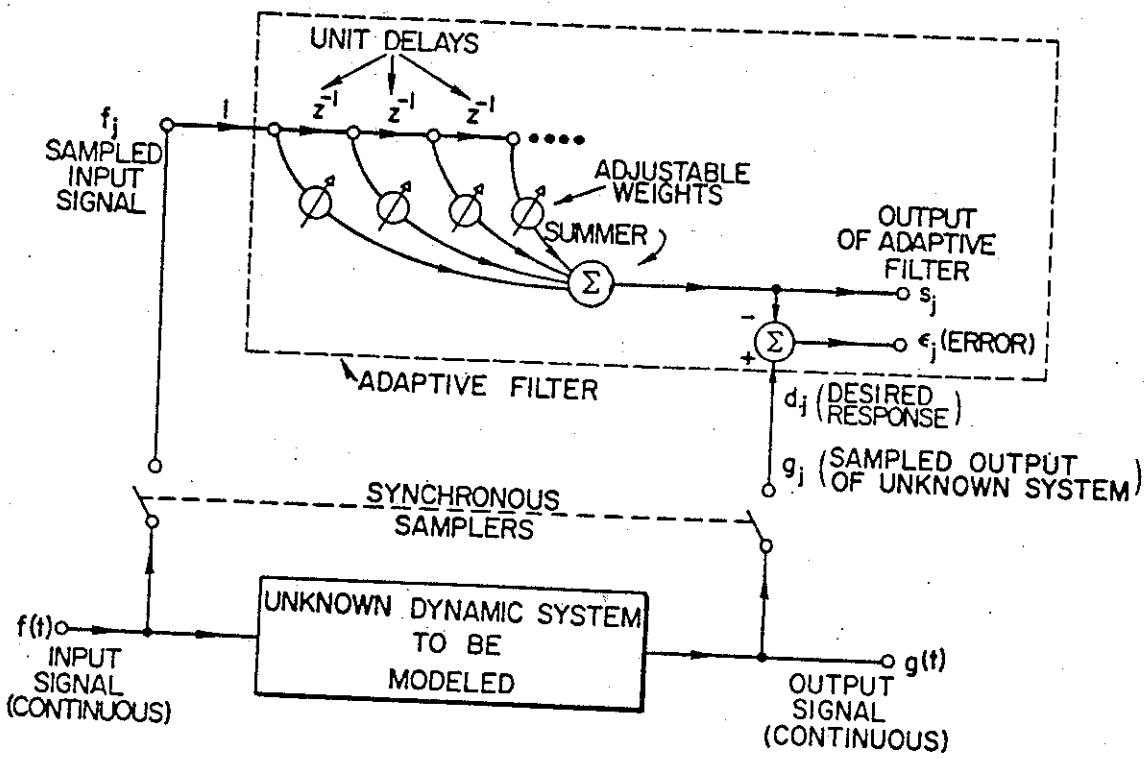
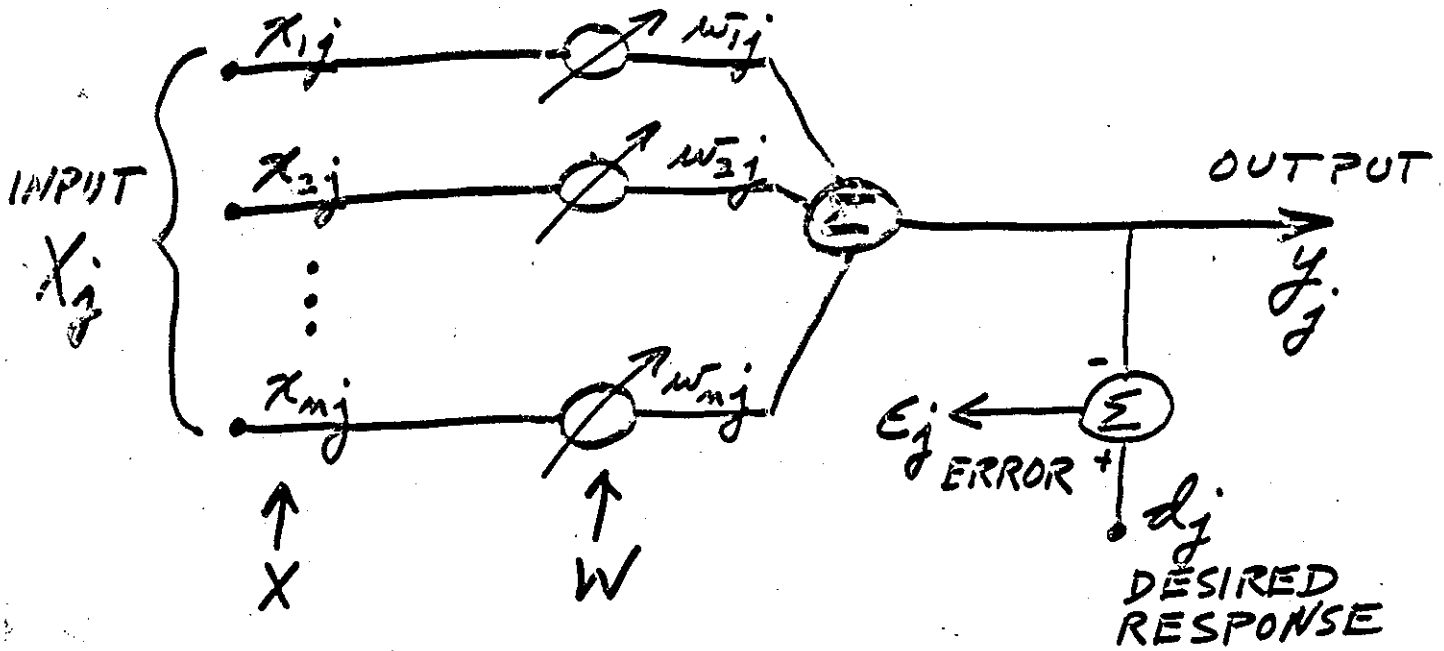


Fig. 1. Modeling an unknown system by a discrete adaptive filter.



$$y_j = X_j^T W = W^T X_j$$

$$\epsilon_j = d_j - X_j^T W$$

$$\epsilon_j^2 = d_j^2 - 2 d_j X_j^T W + W^T X_j X_j^T W$$

$$MSE = \xi \triangleq E[\epsilon_j^2] = E[d_j^2] - 2P^T W + W^T R W$$

$$\nabla = \frac{\partial \xi}{\partial W} = -2P + 2RW$$

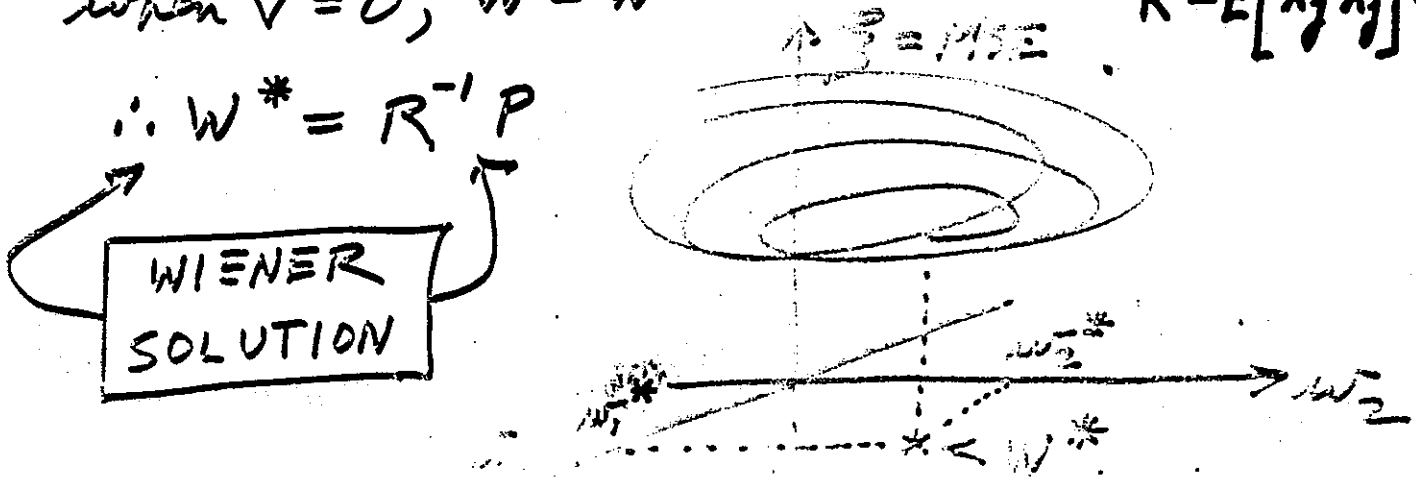
when $\nabla = 0$, $W = W^*$

$$\therefore W^* = R^{-1} P$$

WIENER SOLUTION

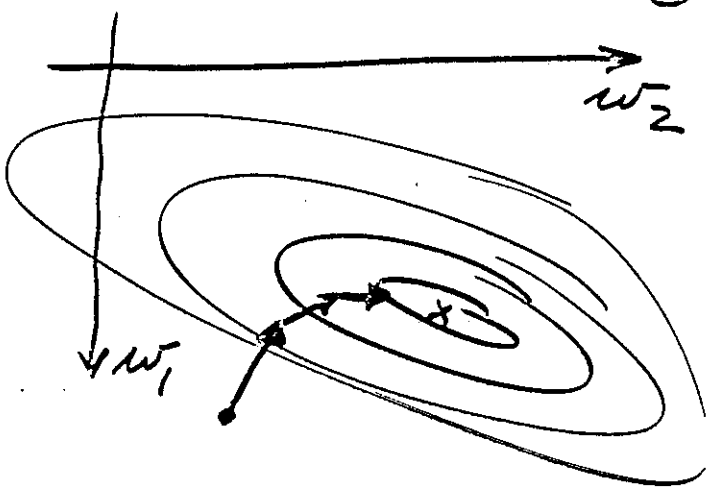
$$P \triangleq E\{d_j X_j\} \quad (n \times 1)$$

$$R \triangleq E[X_j X_j^T] \quad (n \times n)$$



STEEPEST DESCENT:

$$W_{j+1} = W_j - \mu \nabla_j$$



TRUE GRADIENT:

$$\nabla = \frac{\partial \xi}{\partial W} = -2P + 2RW$$

A GRADIENT ESTIMATE:

$$\hat{\nabla} = \frac{\partial \epsilon_j^2}{\partial W} \neq \frac{\partial E[\epsilon_j^2]}{\partial W} \triangleq \nabla$$

now, $\epsilon_j = d_j - X_j^T W$

$$\hat{\nabla} = 2\epsilon_j \frac{\partial \epsilon_j}{\partial W} = 2\epsilon_j(-X_j) = -2\epsilon_j X_j$$

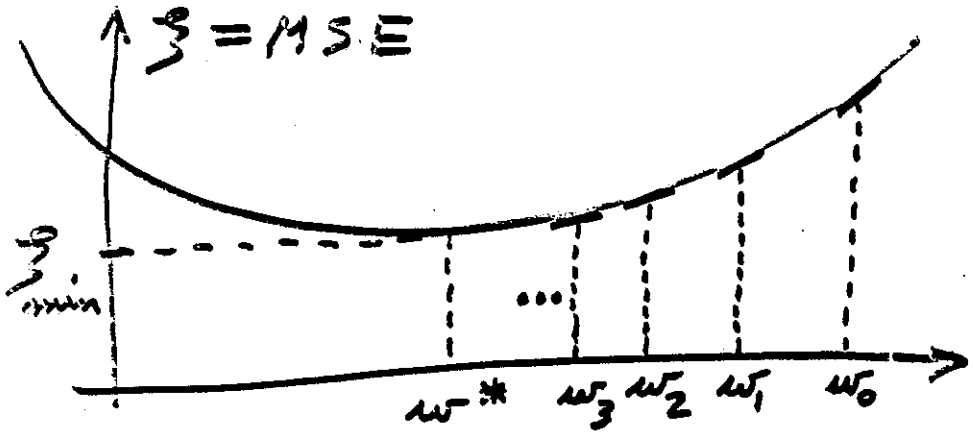
$$E[\hat{\nabla}] = -2E[\epsilon_j X_j] = -2E[d_j X_j - X_j X_j^T W] = -2P + 2RW = \nabla$$

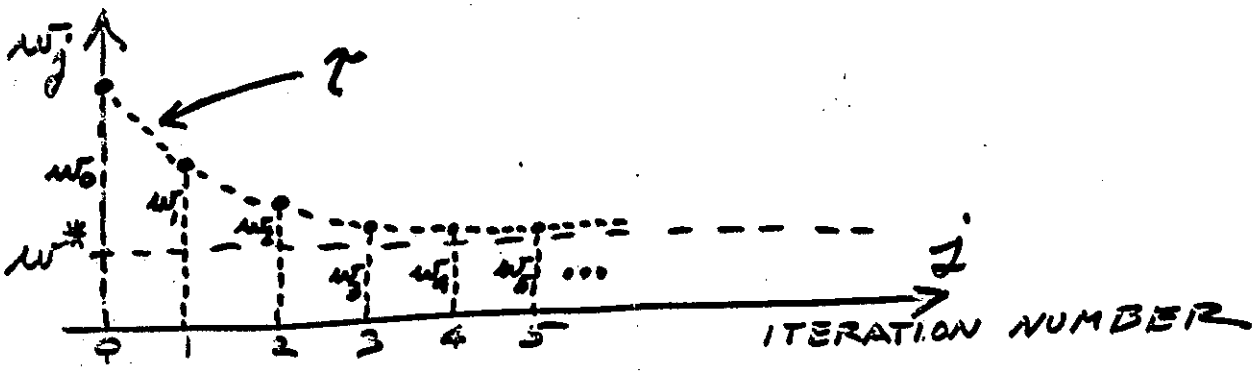
$\therefore$ grad. estimate is unbiased!

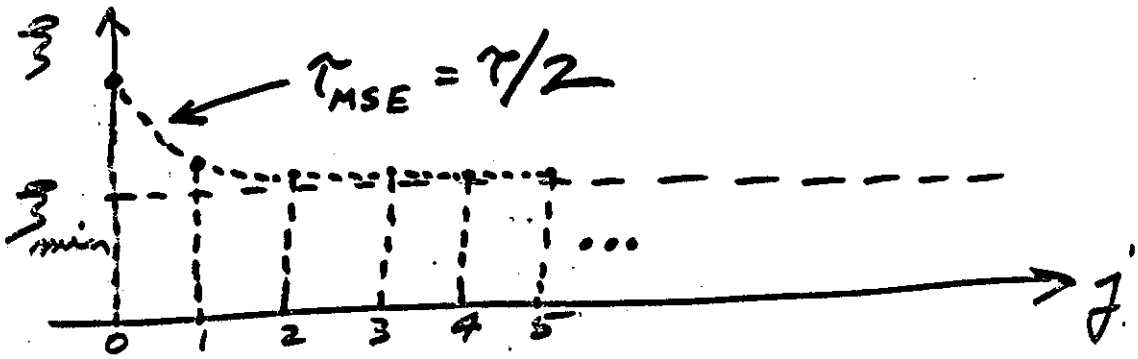
LMS ALGORITHM (Least mean square error) OF WIDROW and HOFF (1959):

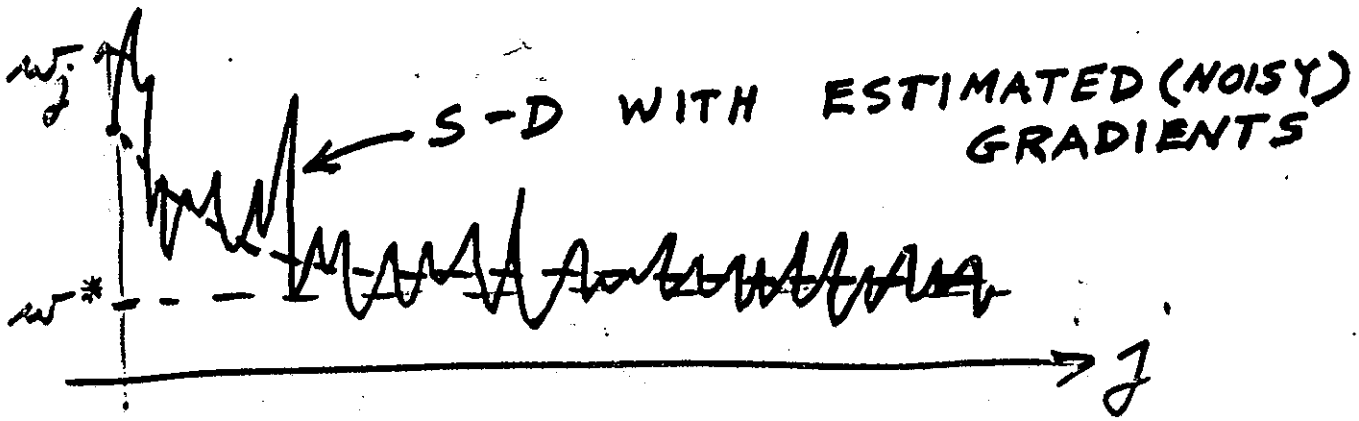
$$W_{j+1} = W_j - \mu \hat{\nabla}_j \leftarrow \text{S-D using estimated gradient}$$

$$\boxed{W_{j+1} = W_j + 2\mu \epsilon_j X_j \quad \text{and} \quad \epsilon_j = d_j - X_j^T W_j} \leftarrow \text{LMS}$$


$\mathcal{E} = MSE$
$\mathcal{E}_{min}$
w^*
w_3
w_2
w_1
w_0


w_j
τ
w_0
w^*
j
0
1
2
3
4
5
ITERATION NUMBER


$\mathcal{E}$
$\tau_{MSE} = \tau/2$
$\mathcal{E}_{min}$
0
1
2
3
4
5
j


w_j
S-D WITH ESTIMATED (NOISY) GRADIENTS
w^*
j

CONDITION FOR CONVERGENCE (STABILITY) OF THE MEAN OF THE WEIGHT VECTOR:

$$\lim_{j\to\infty}\left[I - 2\mu\Lambda\right]^j = 0, \text{ ie}$$

$$\frac{1}{\lambda_{max}} > \mu > 0.$$

SUFFICIENT (BUT NOT NECESSARY), EASY TO APPLY, CONVERGENCE OF THE MEAN:

$$\frac{1}{\text{TRACE } R} > \mu > 0$$

NOTE: CONVERGENCE OF THE MEAN OF THE WEIGHT VECTOR DOES NOT GUARANTEE CONVERGENCE OF ITS VARIANCE OF OTHER MOMENT.

TIME CONSTANTS OF LMS ALGORITHM

$$\tau_p = \frac{1}{2\mu\lambda_p}$$

$$\tau_{p_{MSE}} = \frac{1}{4\mu\lambda_p}$$

MISADJUSTMENT:

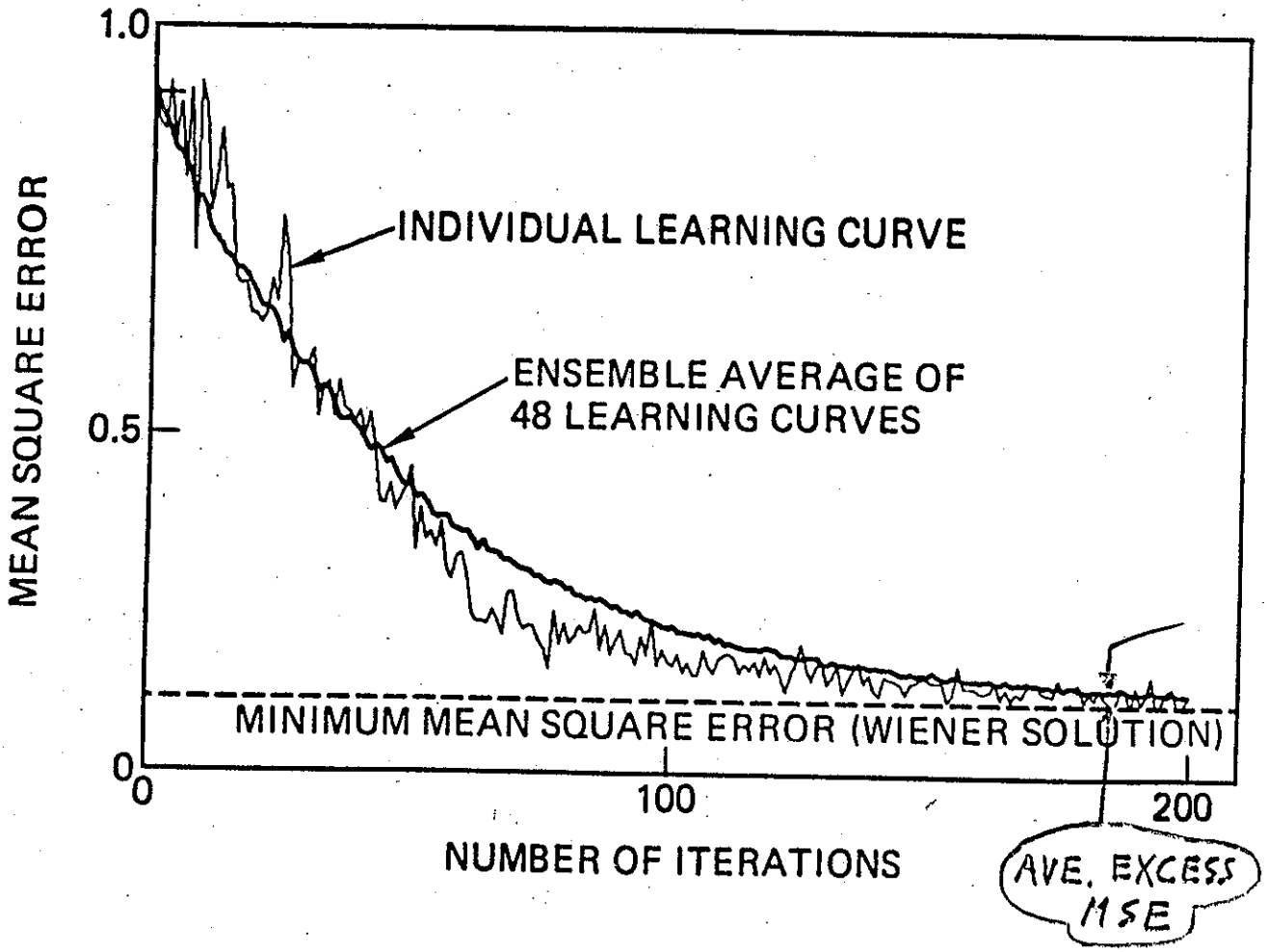
$$M \triangleq \frac{\text{AVE EXCESS MSE}}{\text{MINIMUM MSE}} = \mu\,\text{TRACE}R = \frac{1}{2}\sum_{p=1}^{n}\frac{1}{\tau_p} = \frac{n}{2}\left(\frac{1}{\tau_p}\right)_{AVE}$$

$$M = \frac{n}{4}\left(\frac{1}{\tau_{p_{MSE}}}\right)_{AVE} = \frac{n}{4}\frac{1}{\tau_{MSE}}$$ ← WHEN ALL λ'S ARE EQUAL

EXAMPLE: Let $M = 10\%$ be an "OK" misadj; and let $n = 10$ weights. The question is, how big does τ_{MSE} need to be, and what is the settling time of the adaptive process?

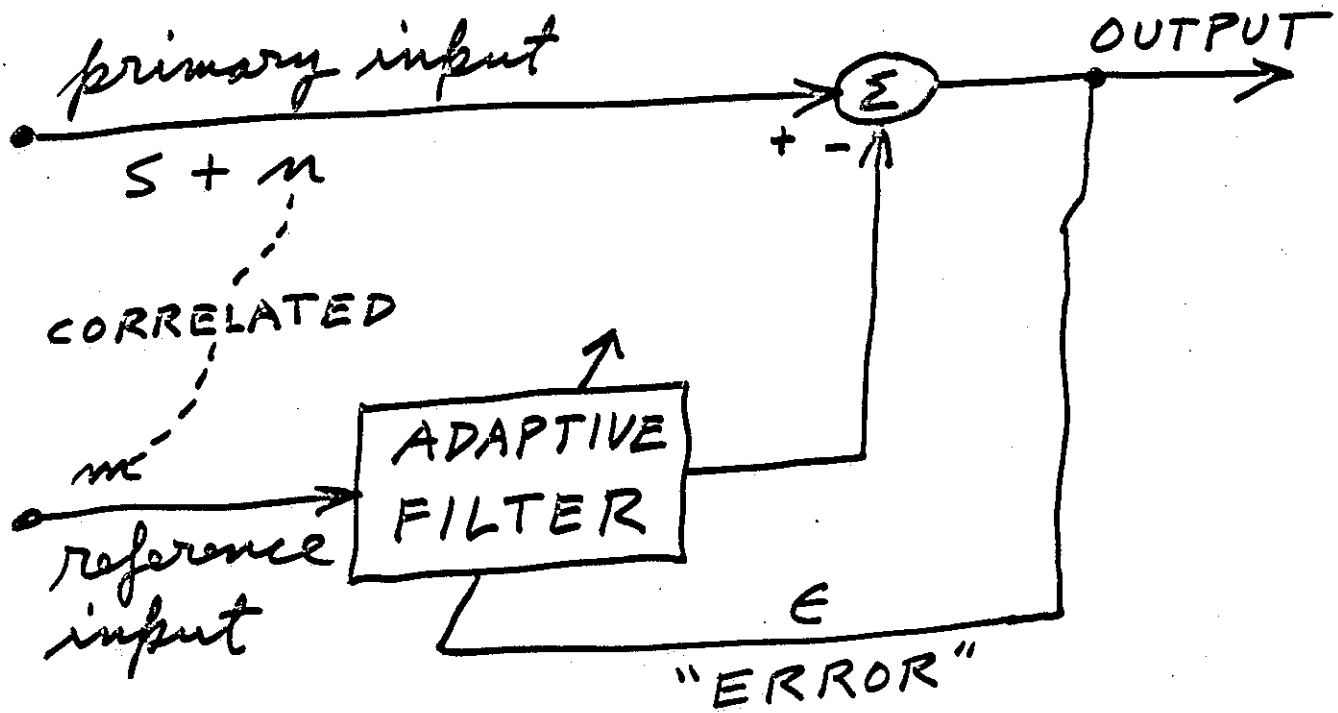
$$M = 0.1 = \frac{10}{4}\frac{1}{\tau_{MSE}} \quad \therefore \tau_{MSE} = 25 \text{ iterations}$$

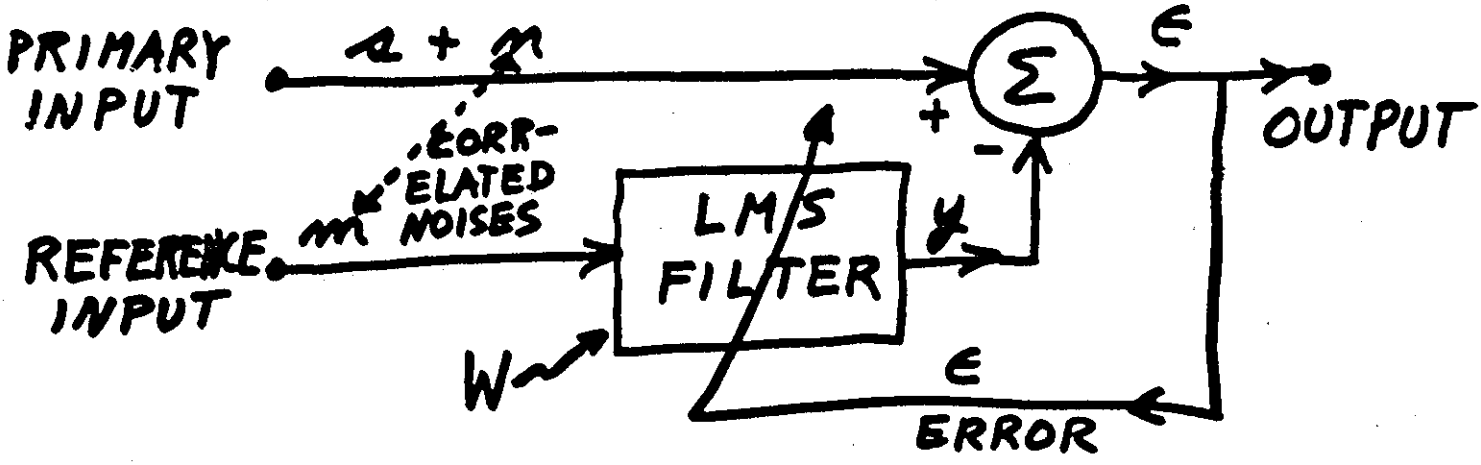
Settling time $\simeq$ 100 iterations = 10X filter length.


1.0
0.5
0
0
100
200
MEAN SQUARE ERROR
INDIVIDUAL LEARNING CURVE
ENSEMBLE AVERAGE OF
48 LEARNING CURVES
MINIMUM MEAN SQUARE ERROR (WIENER SOLUTION)
NUMBER OF ITERATIONS
AVE. EXCESS
MSE

CONVENTIONAL FILTERING

ADAPTIVE NOISE CANCELLING





$$\epsilon = s + n - y$$

$$\epsilon^2 = [s + (n-y)]^2 = s^2 + (n-y)^2 + 2s(n-y)$$

$$E[\epsilon^2] = E[s^2] + E[(n-y)^2] + 2E[s(n-y)]$$

$$E[\epsilon^2] = E[s^2] + E[(n-y)^2].$$

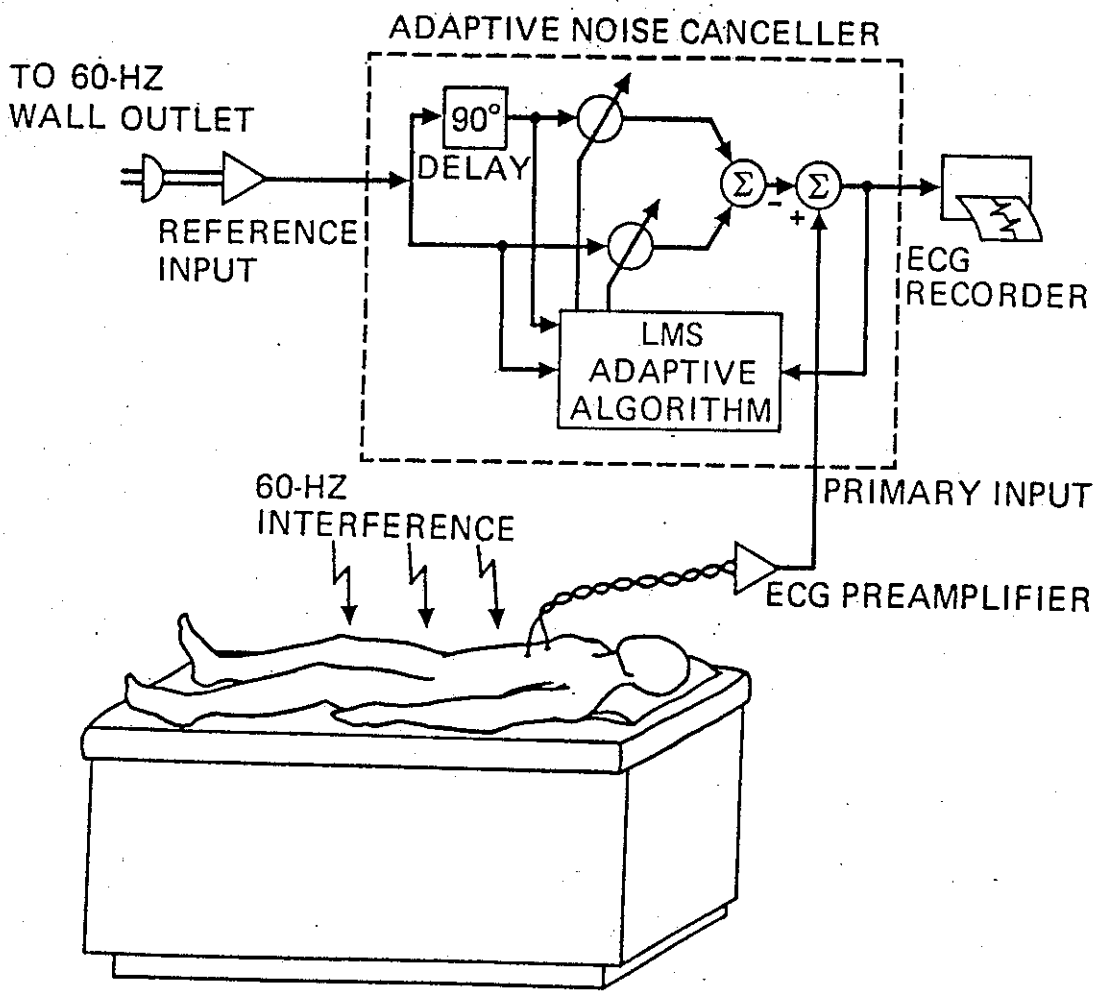
$$\min_W E[\epsilon^2] = E[s^2] + \min_W [E(n-y)^2], \text{ THUS}$$

(1) MINIMIZING $E[\epsilon^2]$ MINIMIZES $E[(n-y)^2]$,

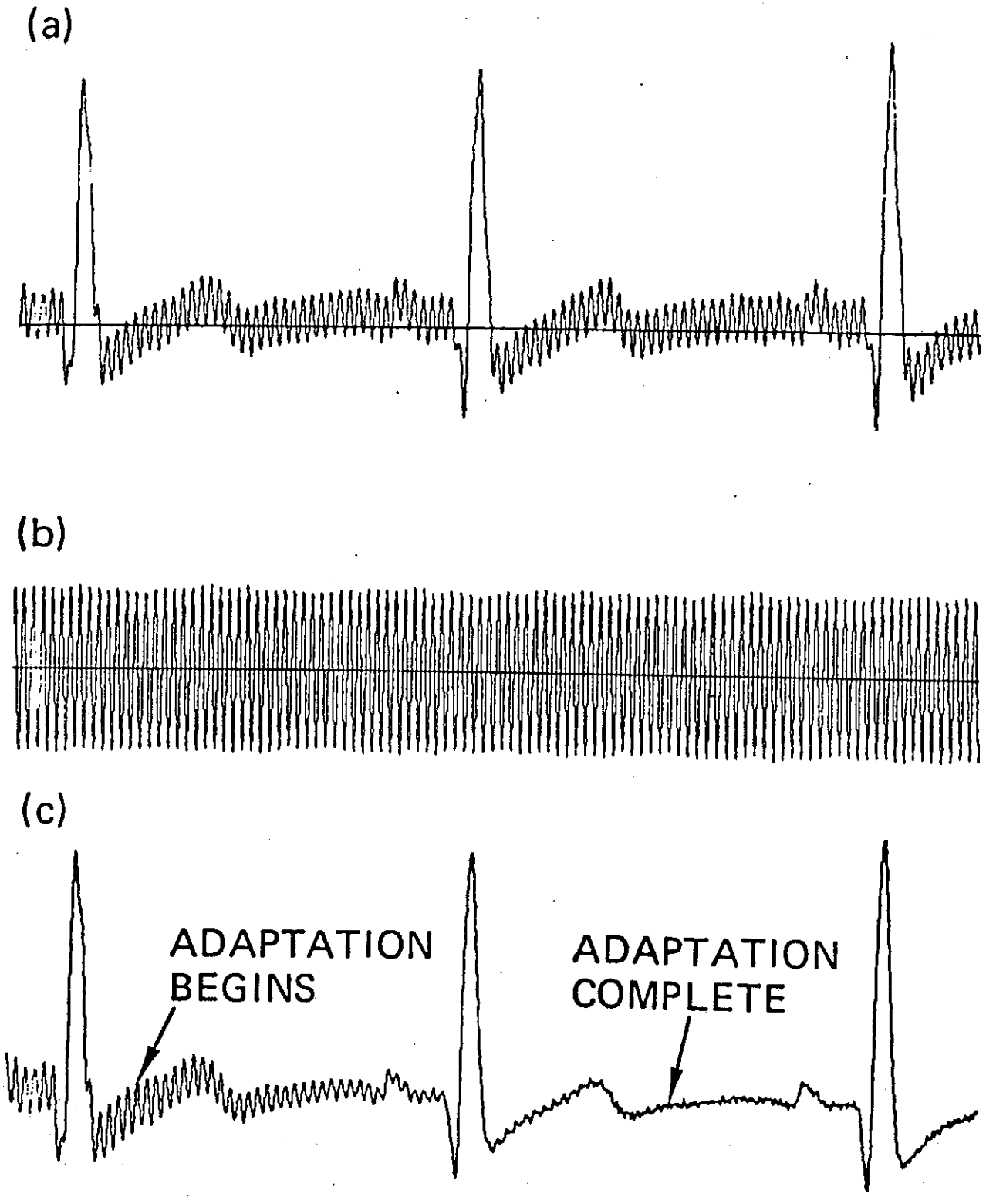
(2) CAUSES y TO BE BEST LEAST SQUARES ESTIMATE OF n,

(3) " ϵ " " " " " " OF s,

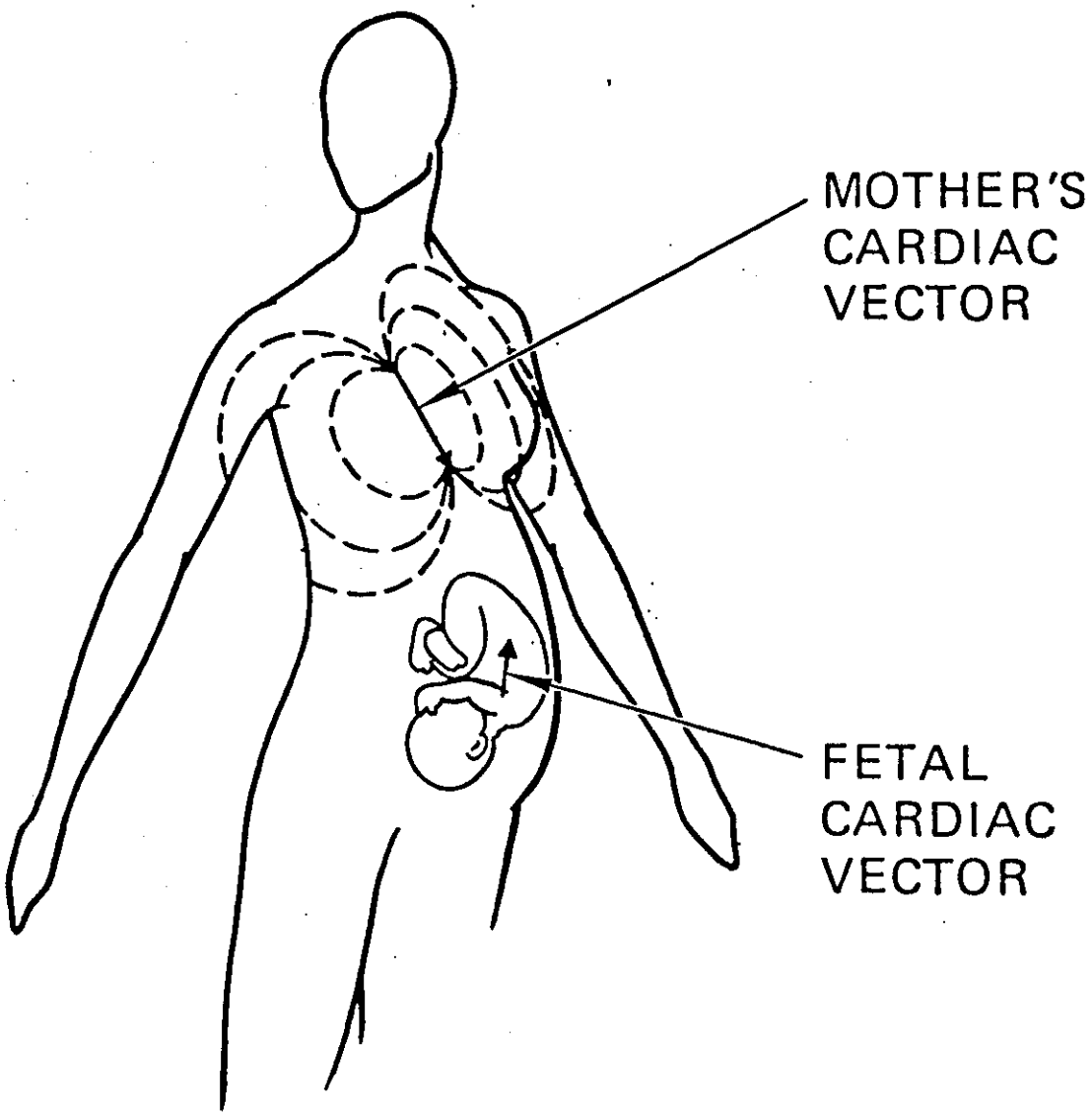
(4) MAXIMIZES OUTPUT SNR WITHOUT SIGNAL DISTORTION



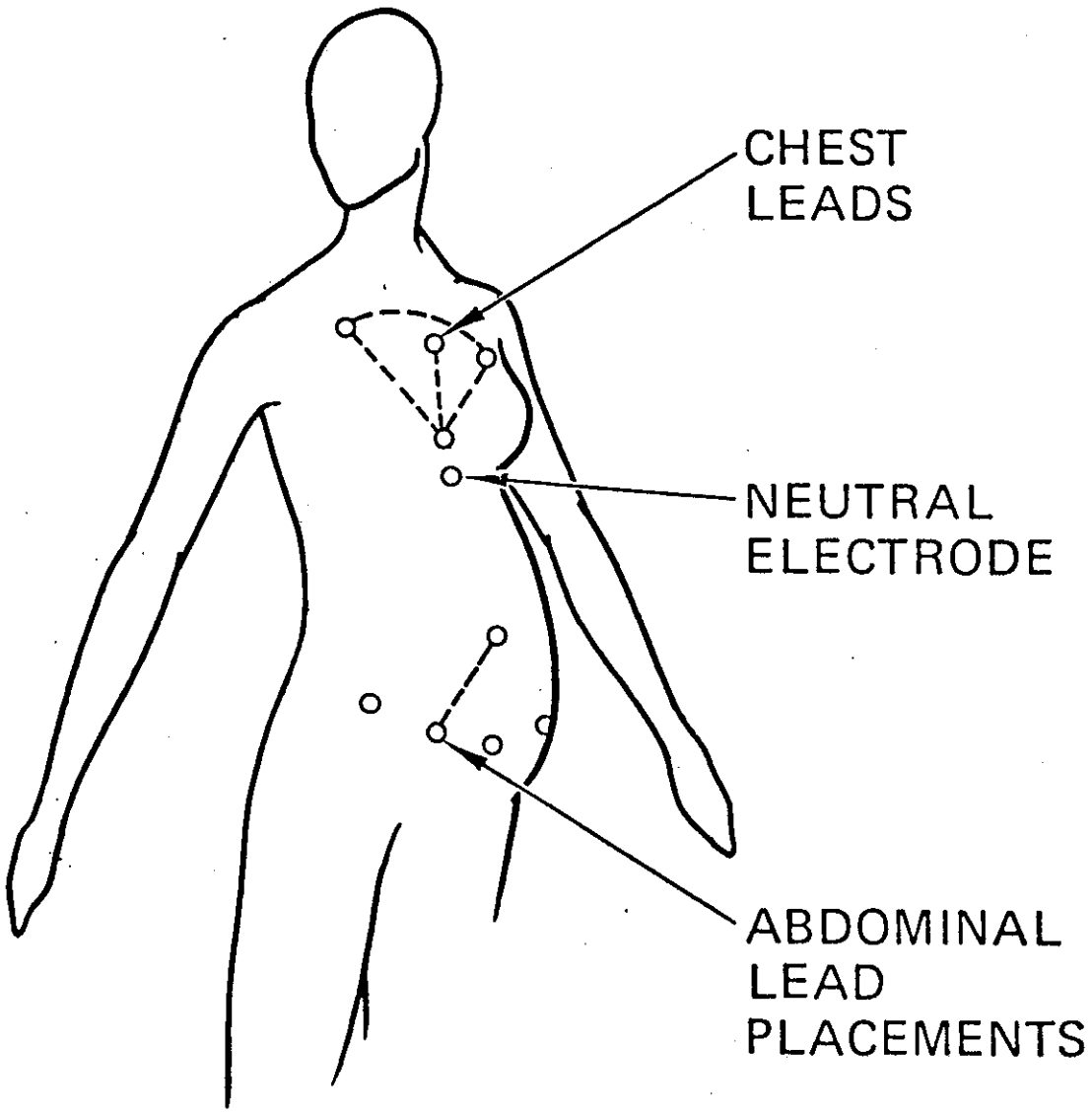
CANCELLING 60Hz INTERFERENCE IN THE EKG

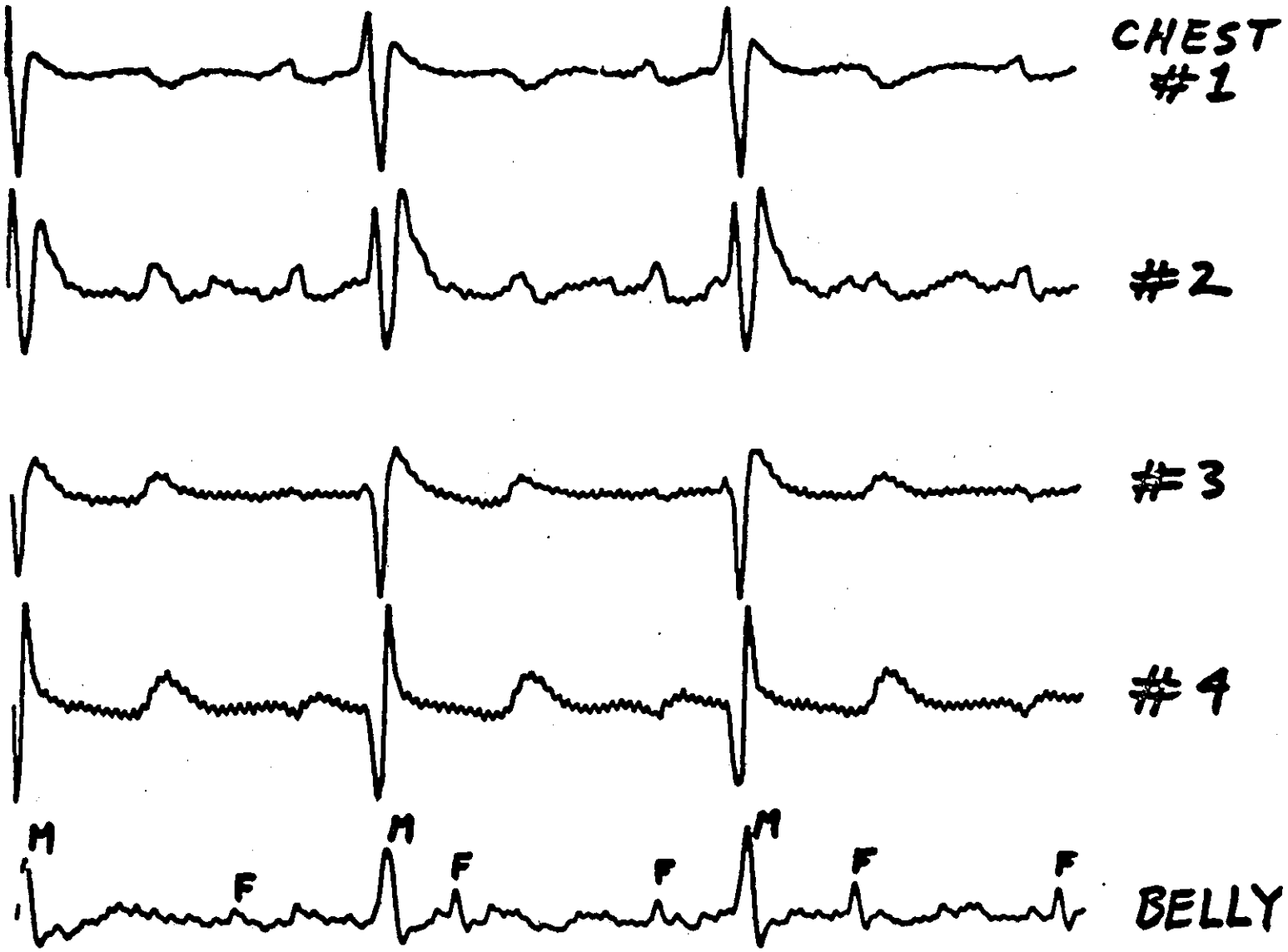

(a)
(b)
(c)
ADAPTATION
BEGINS
ADAPTATION
COMPLETE

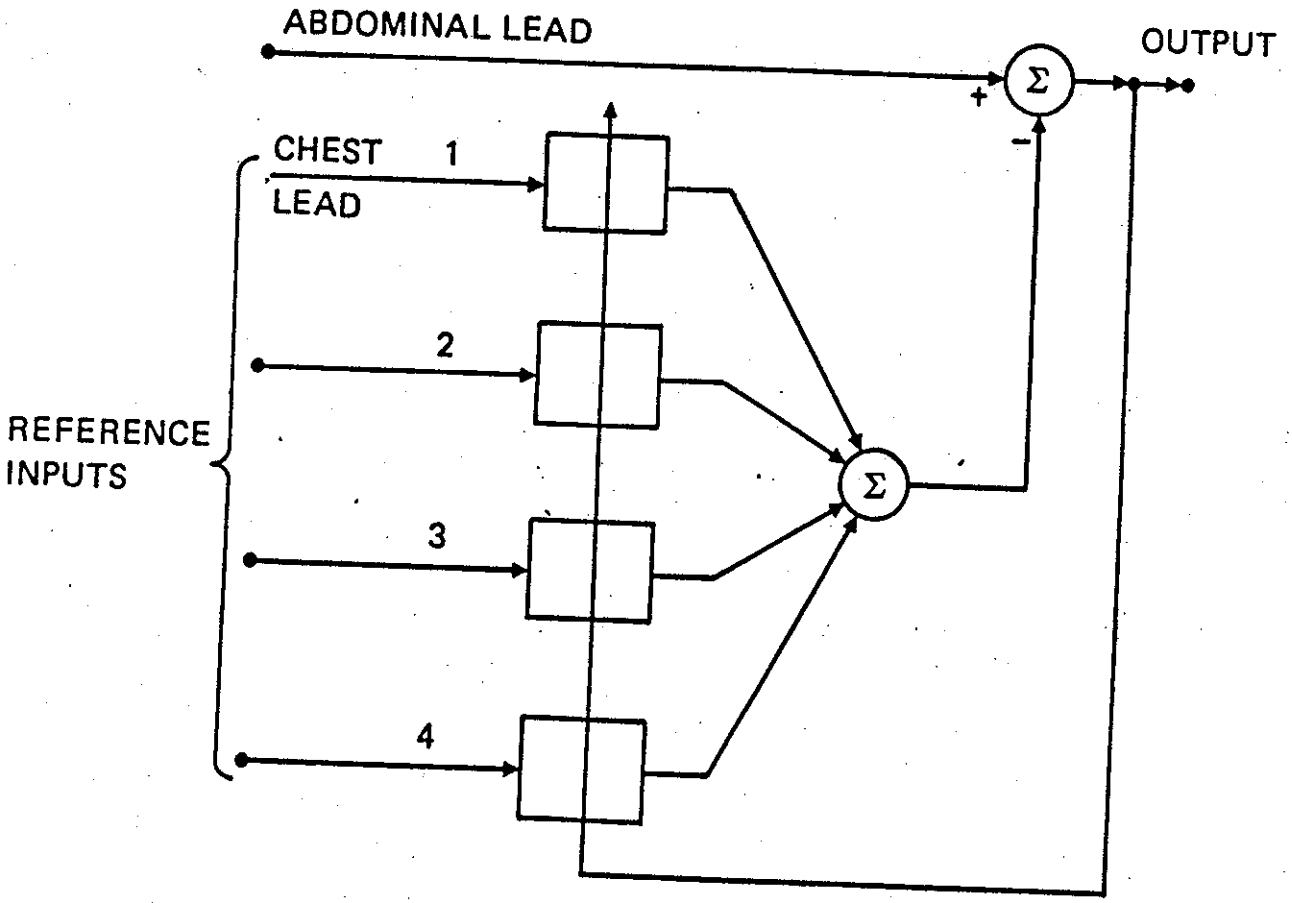
(a)

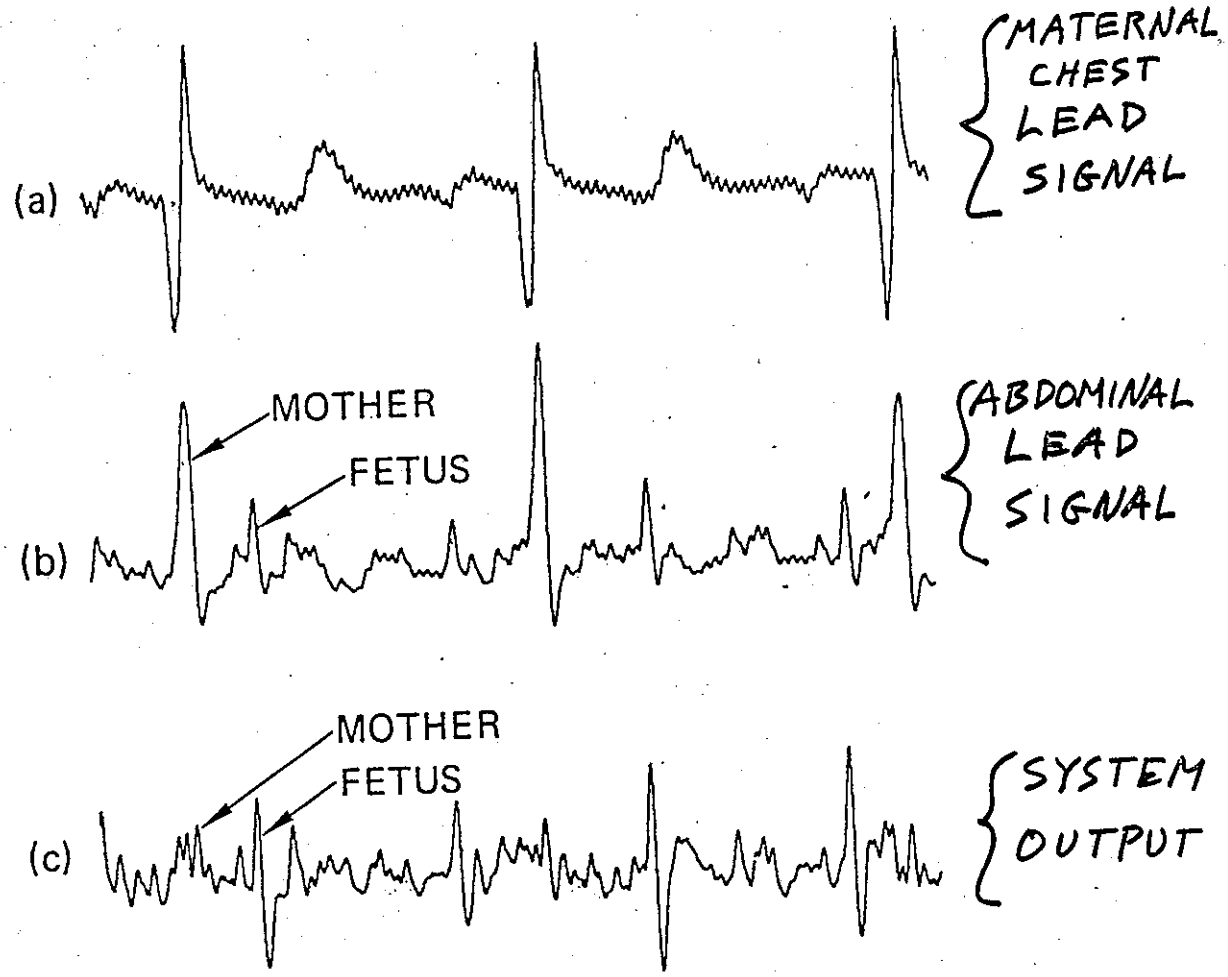


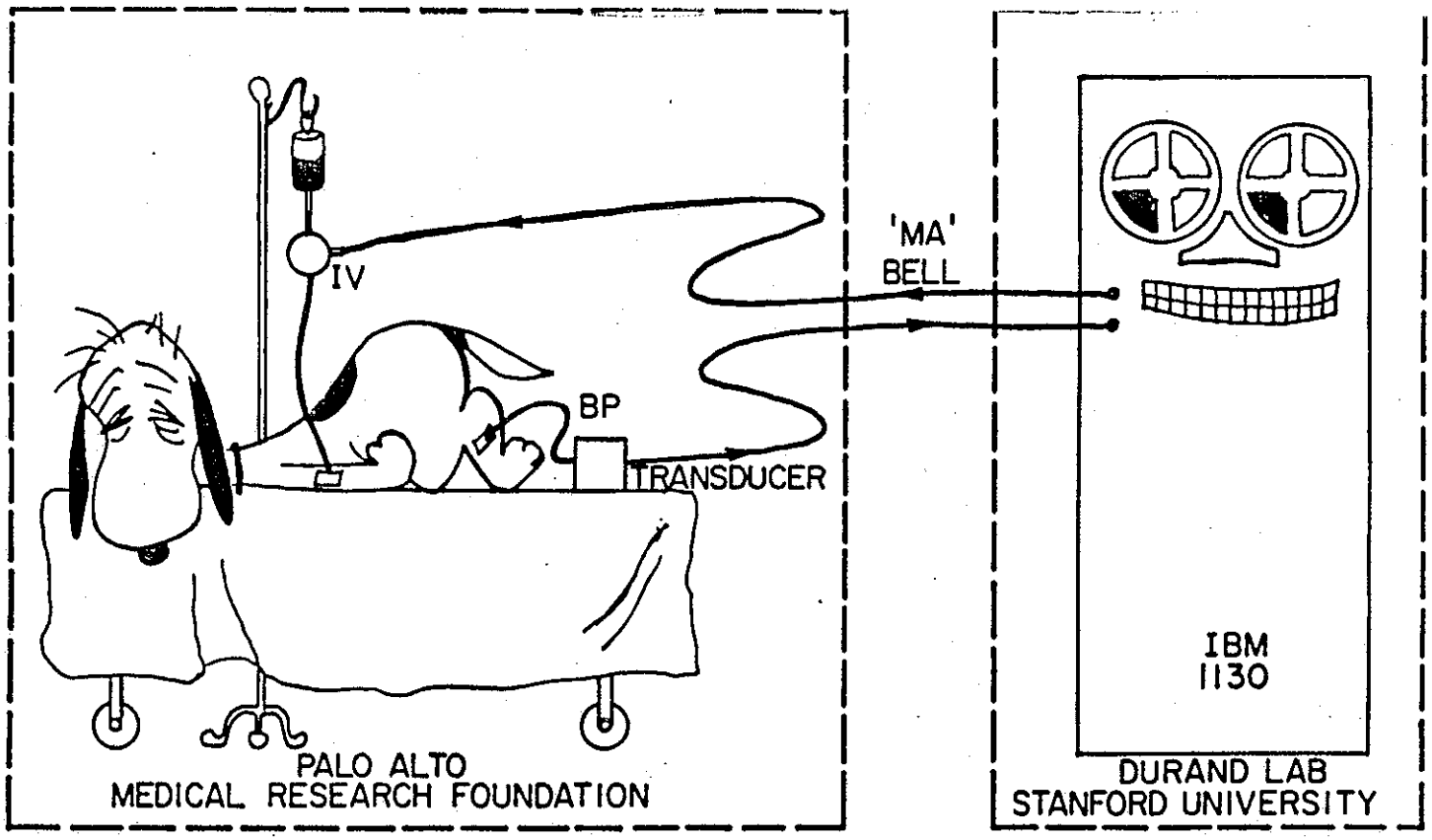
(b)




CHEST #1
#2
#3
#4
M
F
M
F
F
M
F
F
BELLY


ABDOMINAL LEAD
OUTPUT
Σ
+
−
CHEST
LEAD
1
2
3
4
REFERENCE
INPUTS
Σ


(a)
MATERNAL CHEST LEAD SIGNAL
MOTHER
FETUS
(b)
ABDOMINAL LEAD SIGNAL
MOTHER
FETUS
(c)
SYSTEM OUTPUT


IV
'MA'
BELL
BP
TRANSDUCER
IBM
1130
PALO ALTO
MEDICAL RESEARCH FOUNDATION
DURAND LAB
STANFORD UNIVERSITY

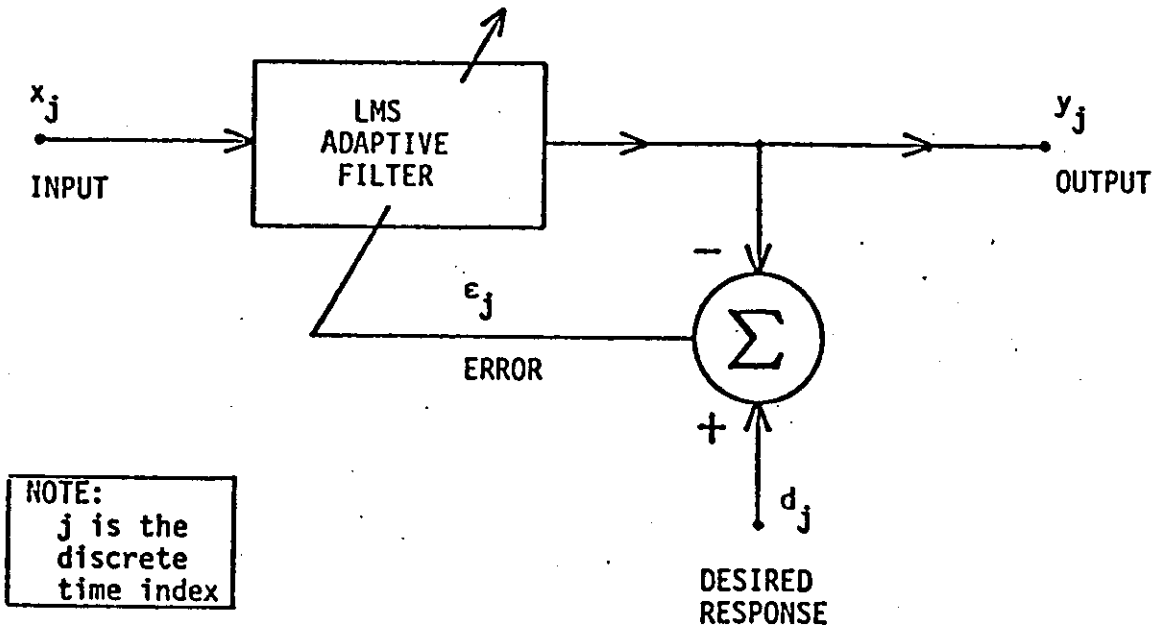


Fig. 1. Symbolic Representation of an Adaptive Transversal Filter Adapted by the LMS Algorithm.

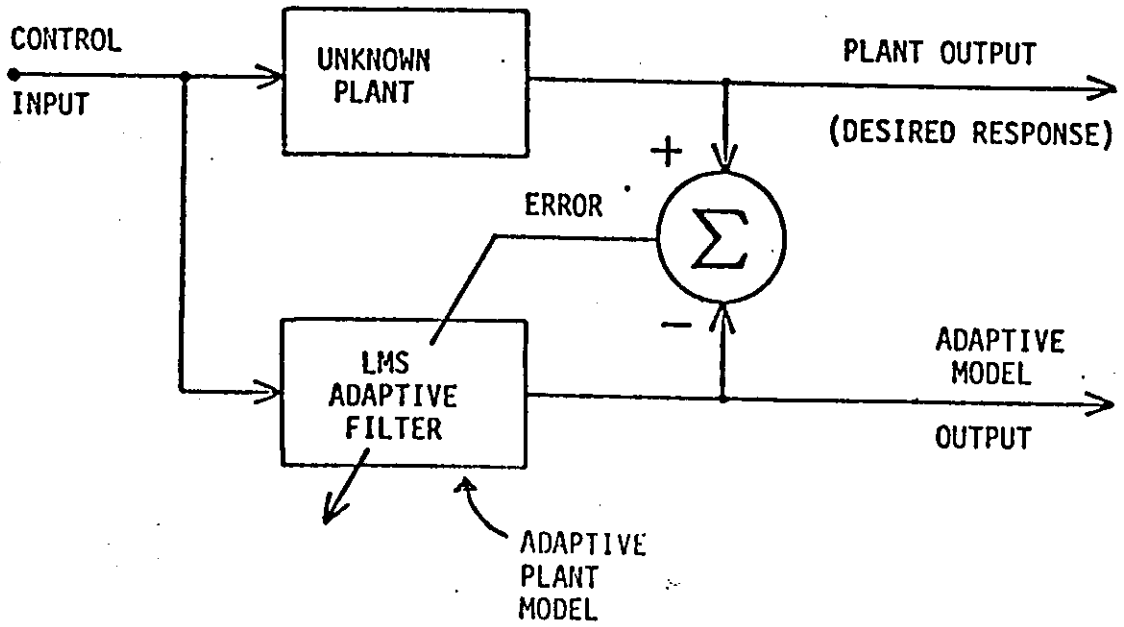


Fig. 2. Modeling an Unknown Plant by Means of an Adaptive Transversal Filter.

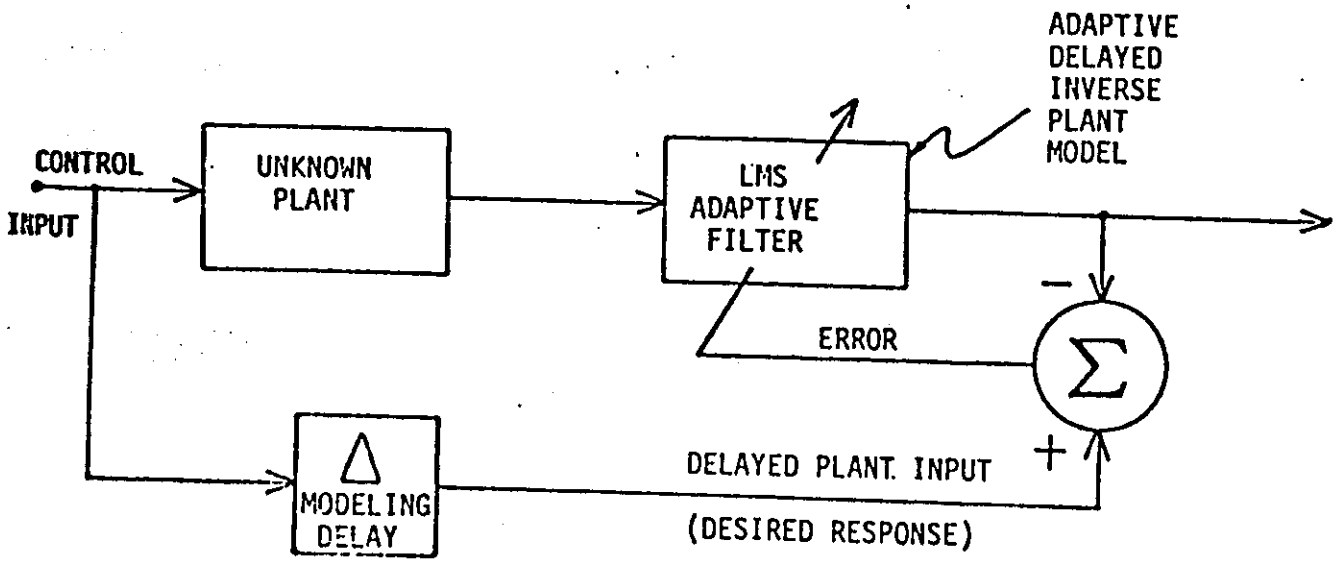


Fig. 4. Delayed Inverse Modeling of an Unknown Plant.

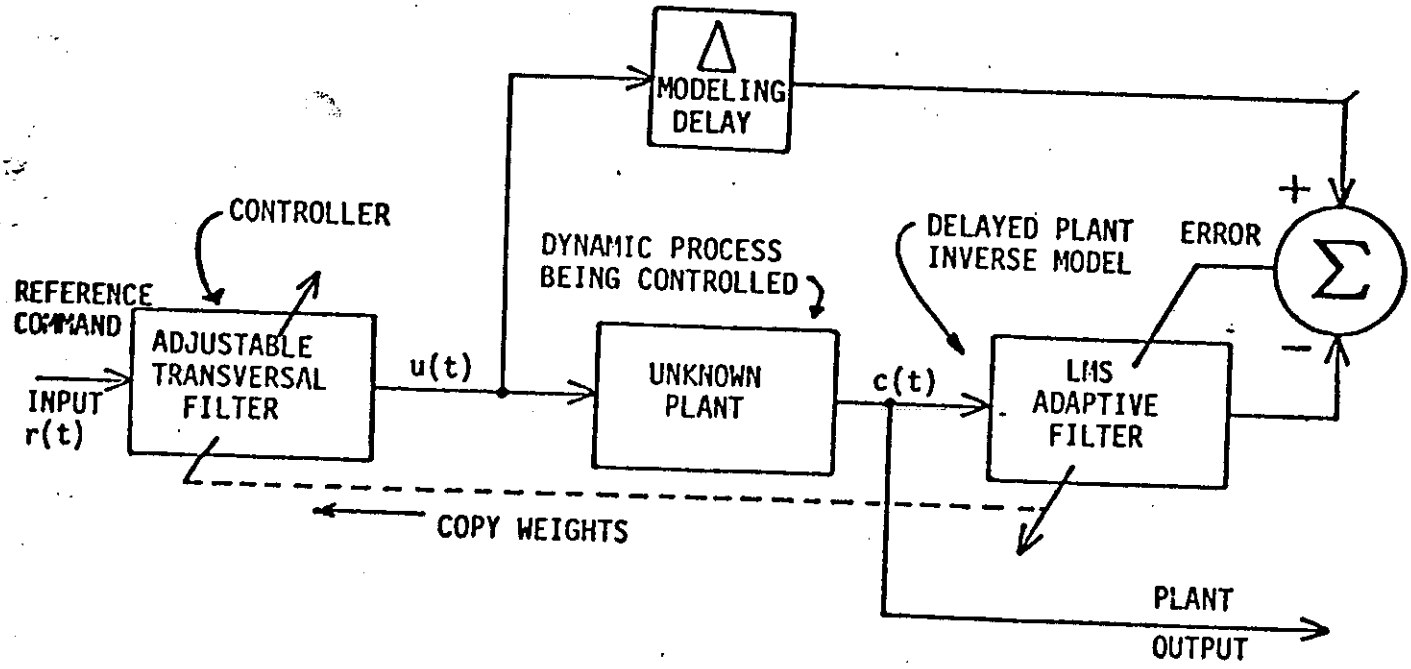


Fig. 5. Adaptive Inverse Model Control System.

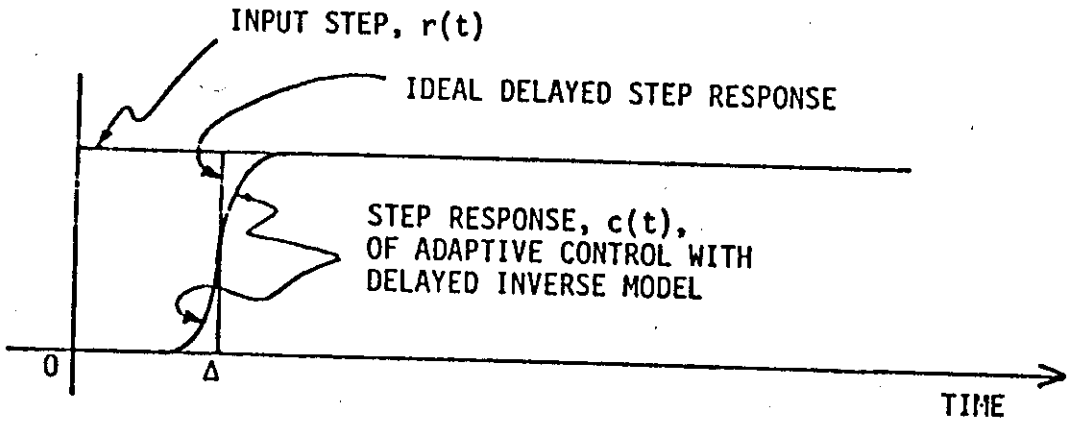


Fig. 6. Comparison of Ideal Step Response vs. Adaptive Delayed Inverse Model Control System Step Response.

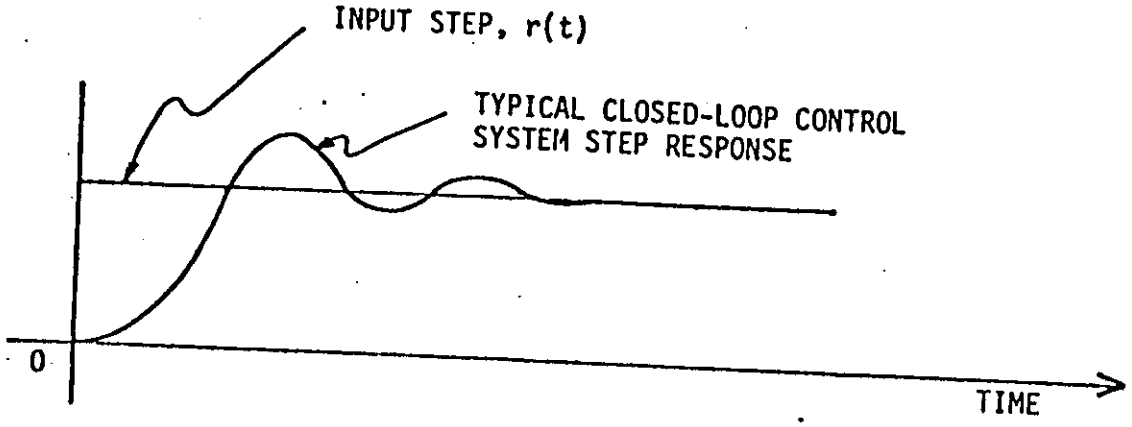


Fig. 7. Step Response of a Conventional Closed-Loop Control System.

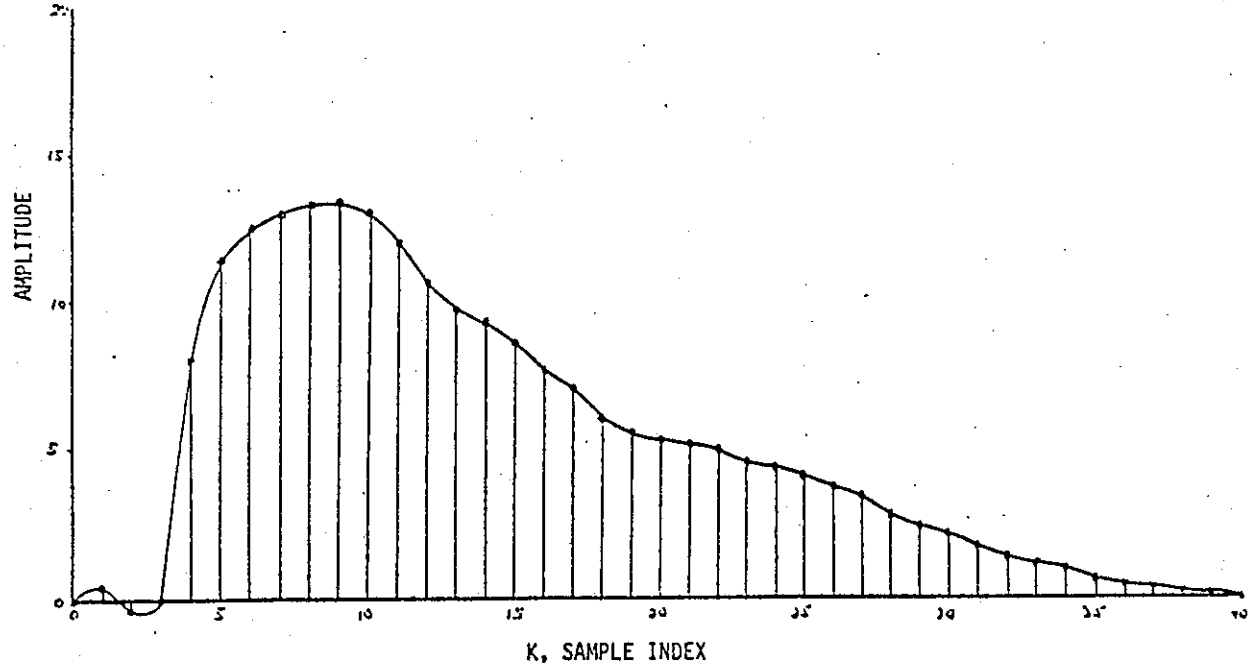


Fig. 2.5. A Discrete Impulse Response Whose Inverse Will Be Sought.

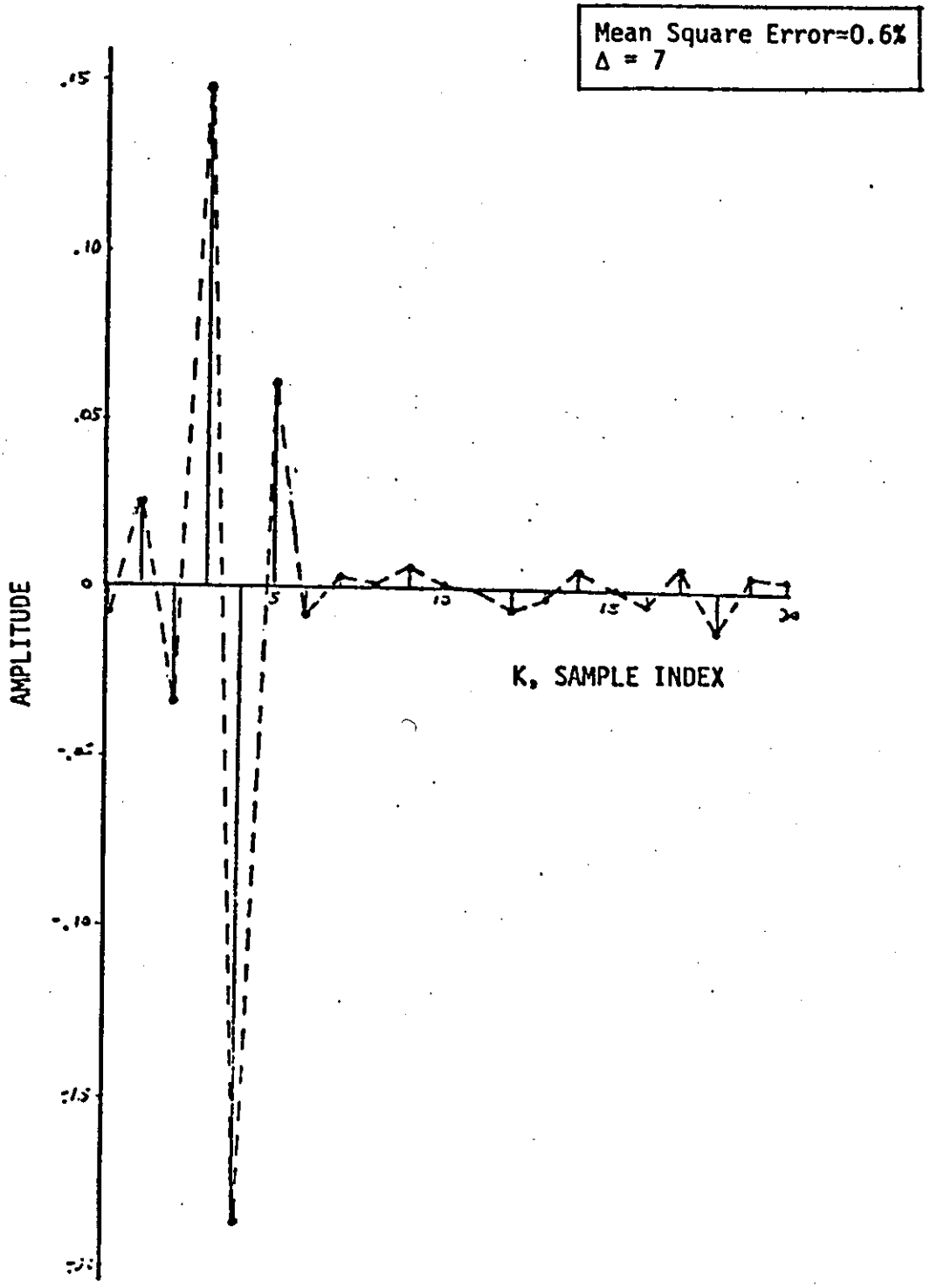


Fig. 2.7. A 21 Weight Delayed Approximate Inverse Impulse Response (7 Units of Delay).

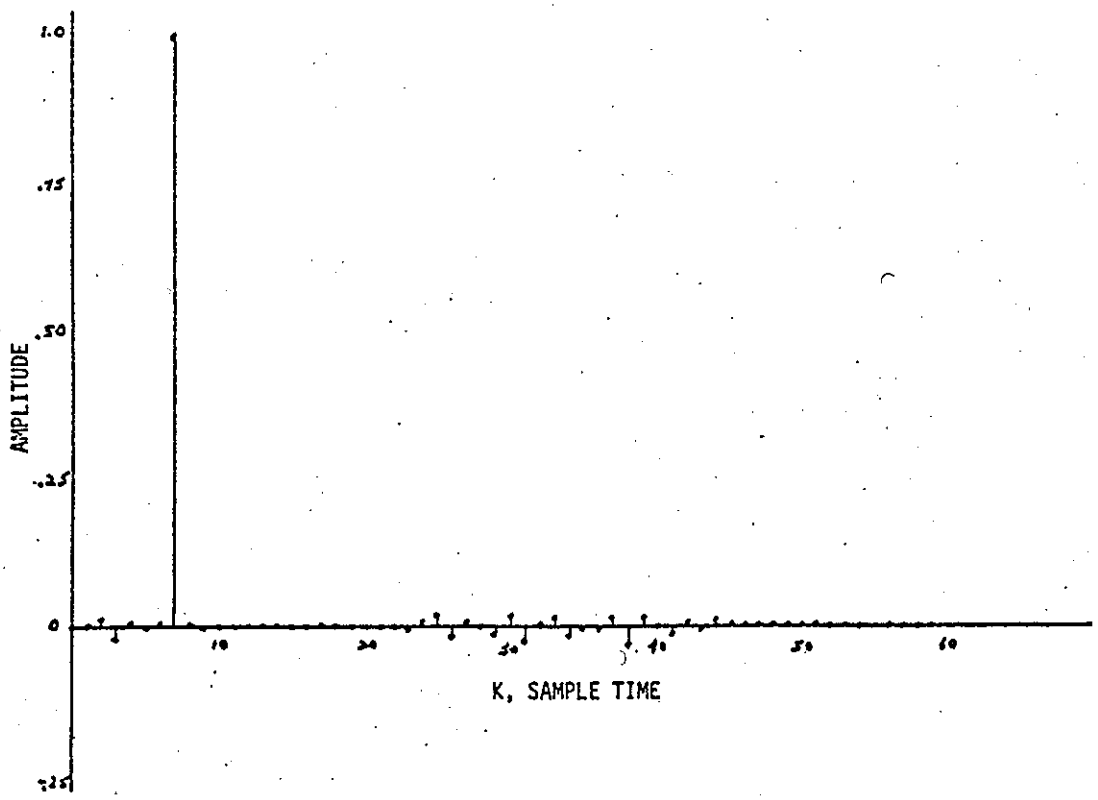


Fig. 2.8. Convolution of the Channel Impulse Response with Its Approximate Inverse

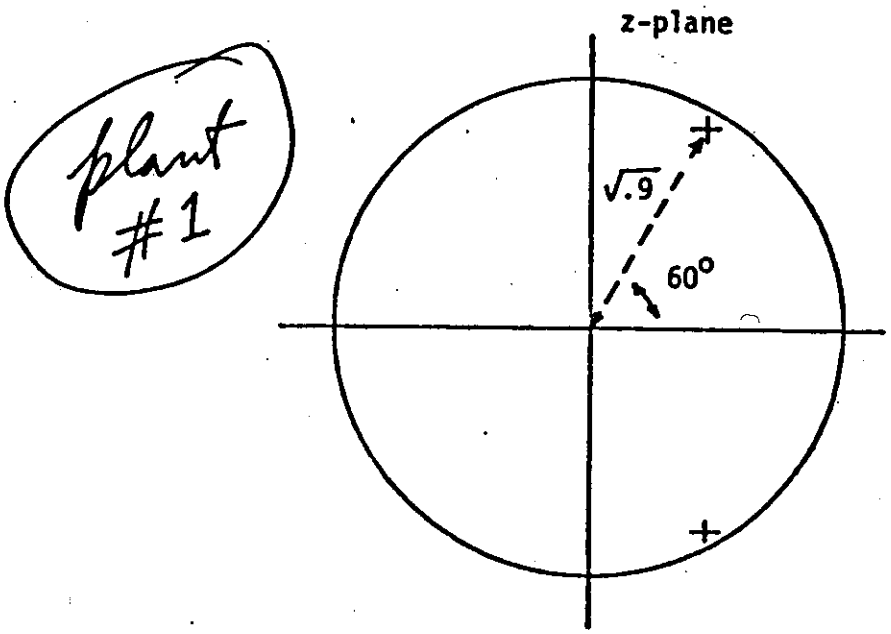


Fig. 10. Pole Locations of Plant #1.

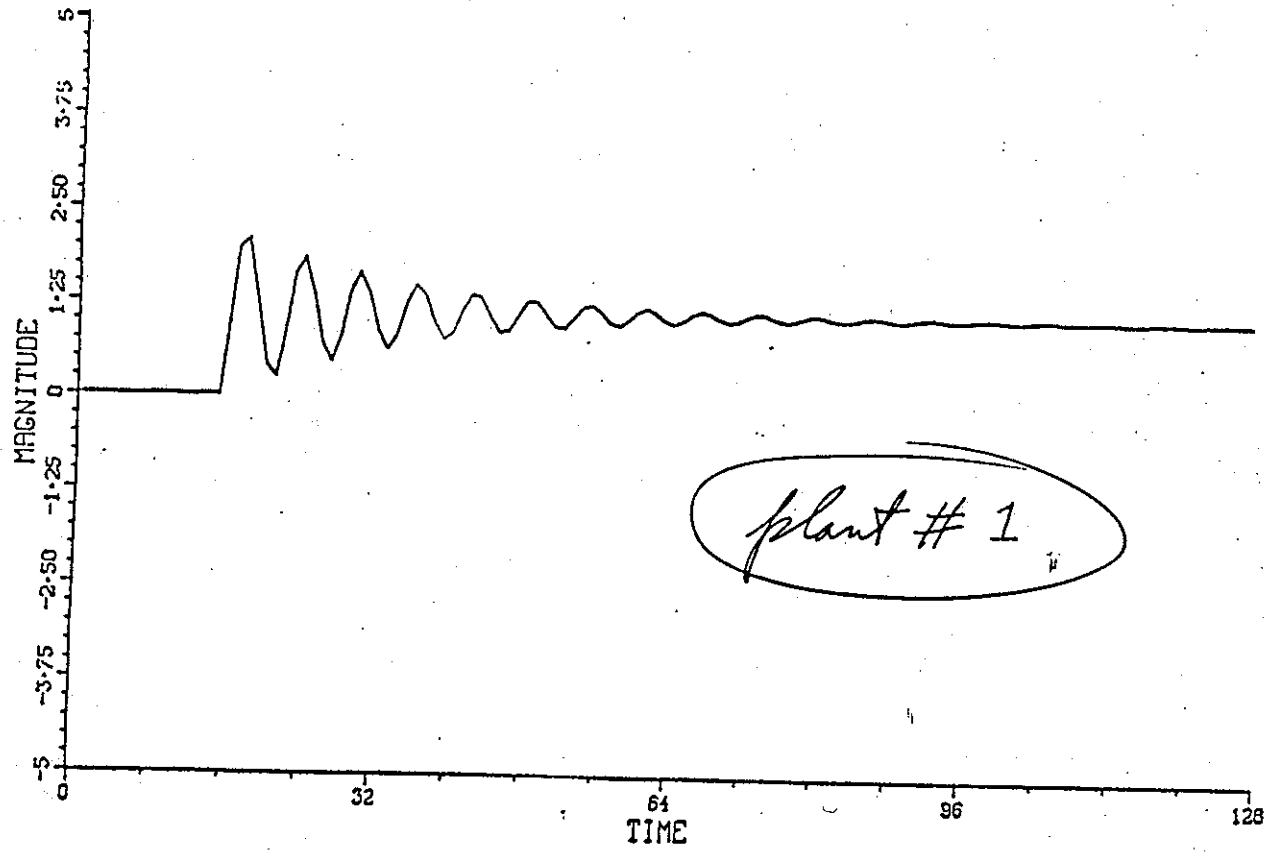


Fig. 11, Step Response of Plant #1.

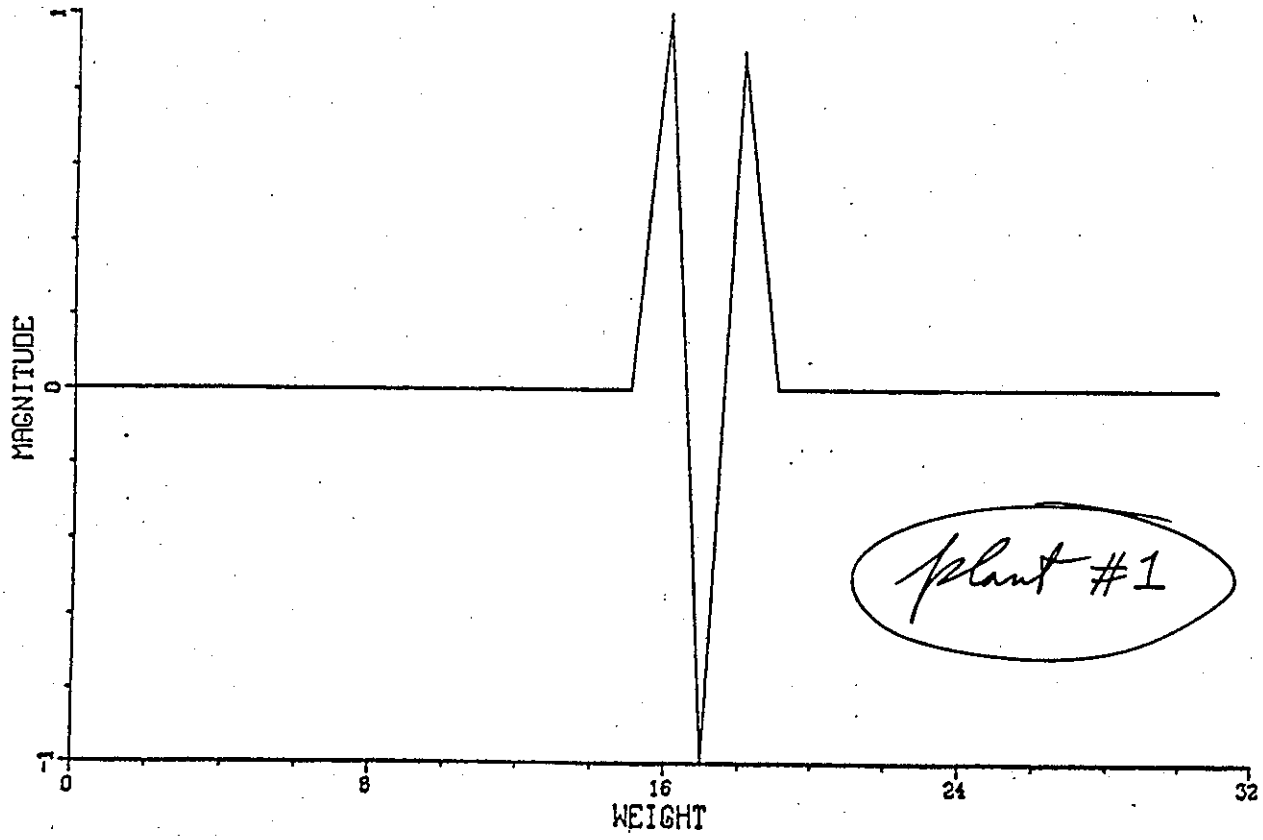


Fig. 12. Impulse Response of Inverse of Plant #1.

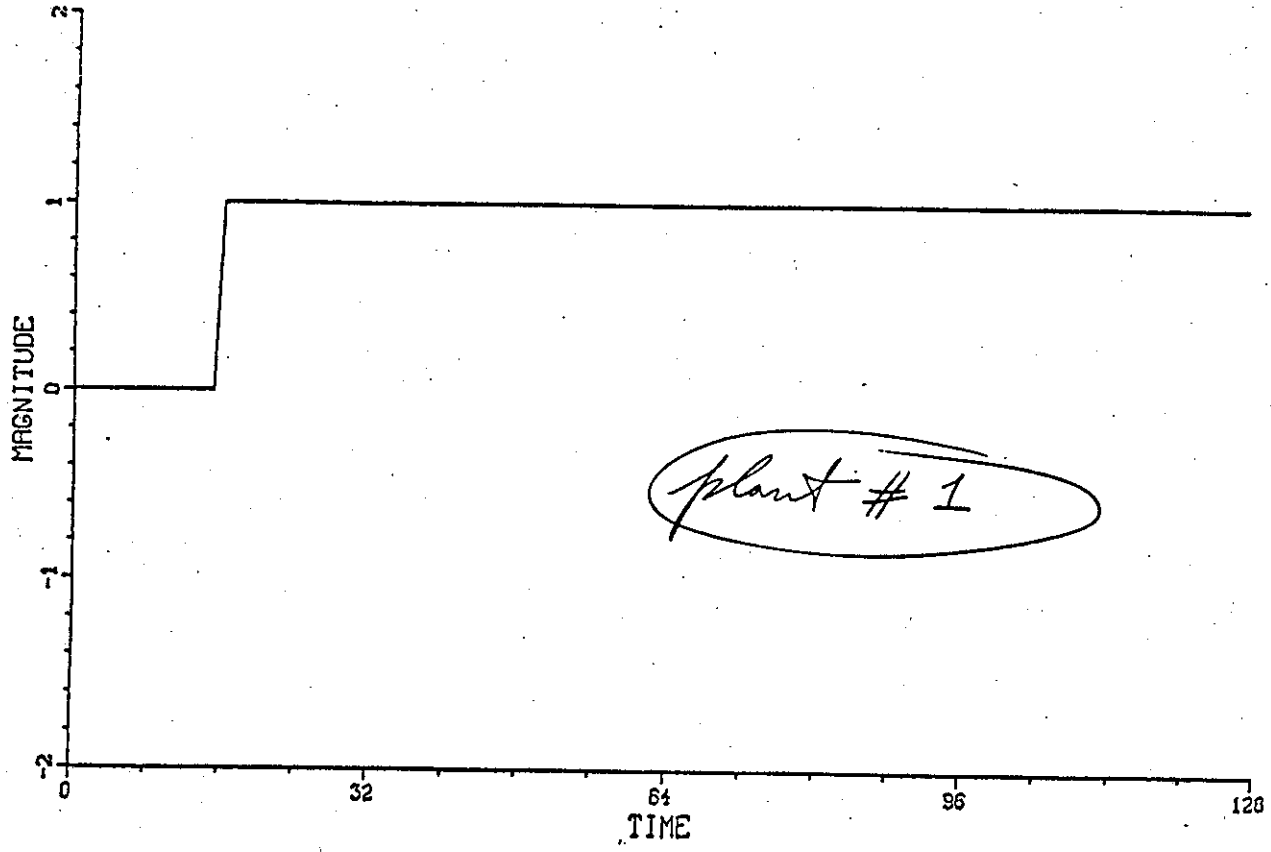


Fig. 13. Step Response of Inverse Model Control with Plant #1.

Proc. IEEE, vol. 78, no. 9, pp 1415-1441, Sept., 1990

30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation

Bernard Widrow Michael A. Lehr

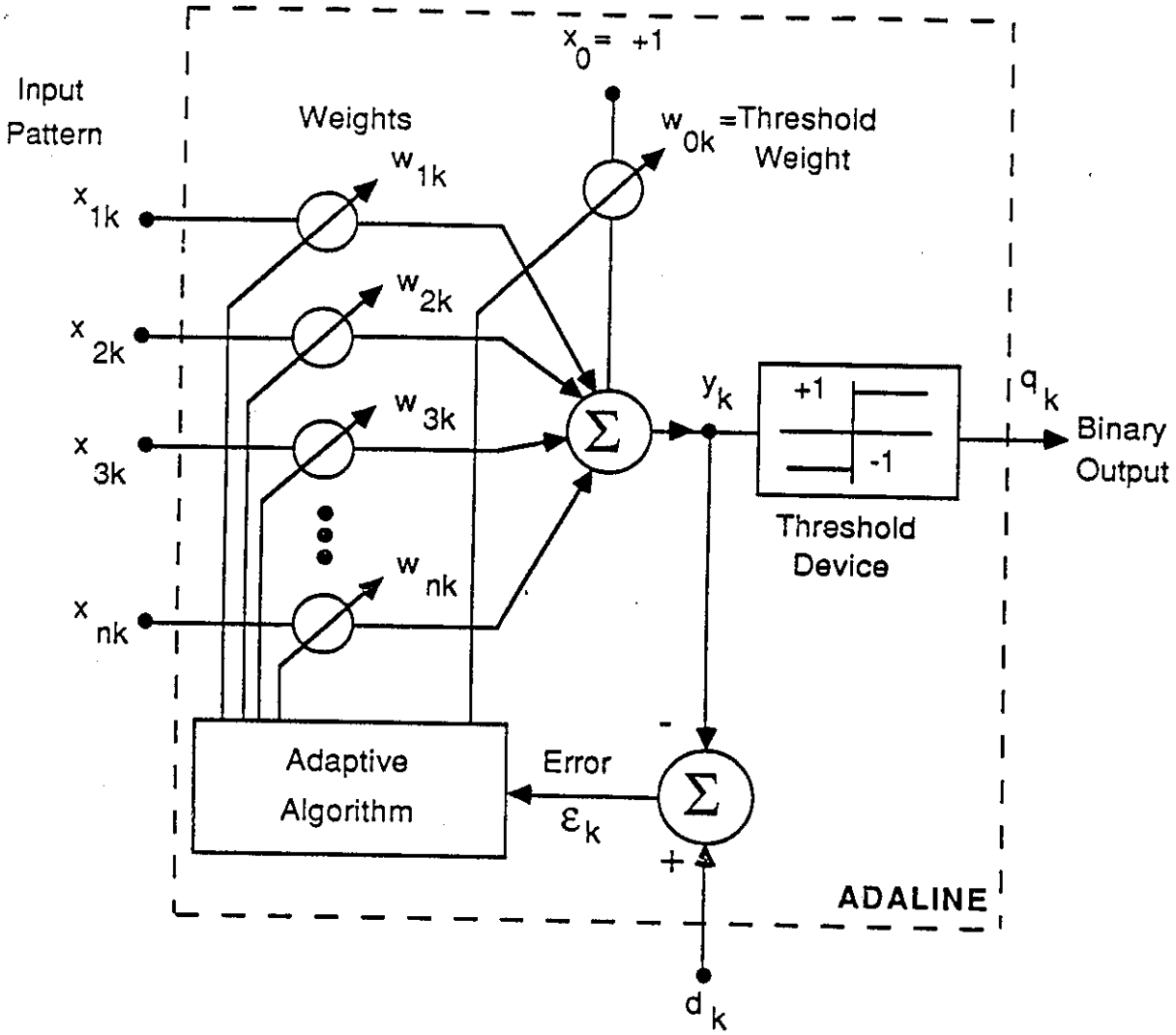
Stanford University Department of Electrical Engineering,

Stanford, CA 94305-4055

Abstract

Fundamental developments in feedforward artificial neural networks from the past thirty years are reviewed. The central theme of this paper is a description of the history, origination, operating characteristics, and basic theory of several supervised neural network training algorithms including the Perceptron rule, the LMS algorithm, three Madaline rules, and the backpropagation technique. These methods were developed independently, but with the perspective of history they can all be related to each other. The concept which underlies these algorithms is the "minimal disturbance principle," which suggests that during training it is advisable to inject new information into a network in a manner which disturbs existing information to the smallest extent possible.

37a



$$X_k = [x_{0k}, x_{1k}, x_{2k}, \ldots, x_{nk}]^T$$
$$= [+1, x_{1k}, x_{2k}, \ldots, x_{nk}]^T$$

$$W_k = [w_{0k}, w_{1k}, w_{2k}, \ldots, w_{nk}]^T$$

$$y_k = X_k^T W_k = W_k^T X_k$$

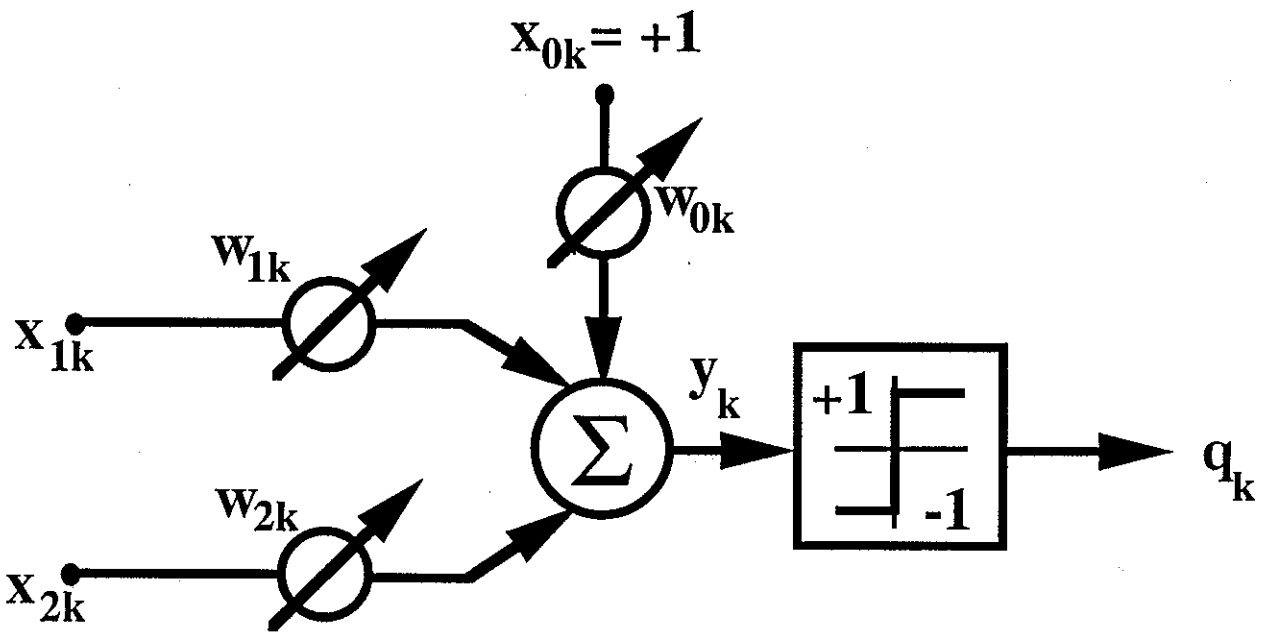
$$\varepsilon_k = d_k - y_k$$

$_k$ is time index

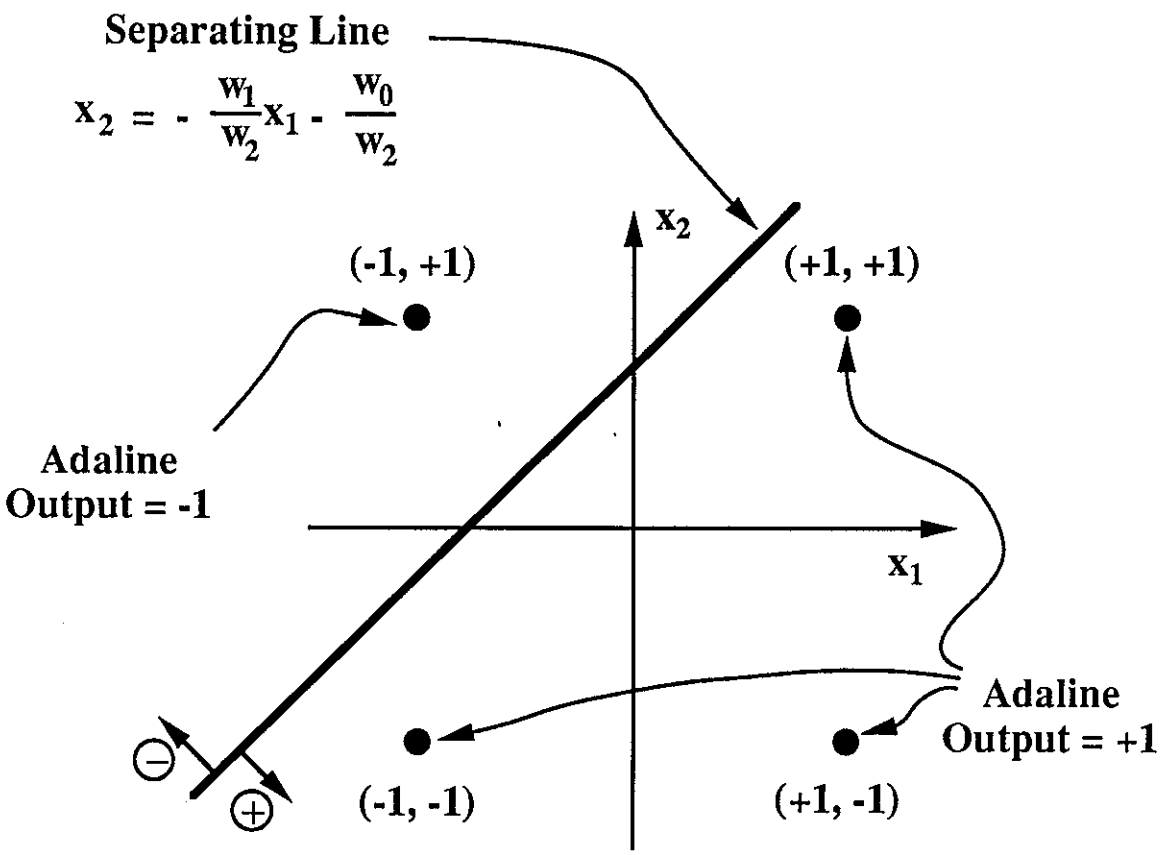
^{T} denotes vector transpose

An adaptive linear neuron (ADALINE).

$$
\begin{aligned}
y &= x_1 w_1 + x_2 w_2 + w_0 = 0 \\
x_2 &= -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}
\end{aligned}
$$

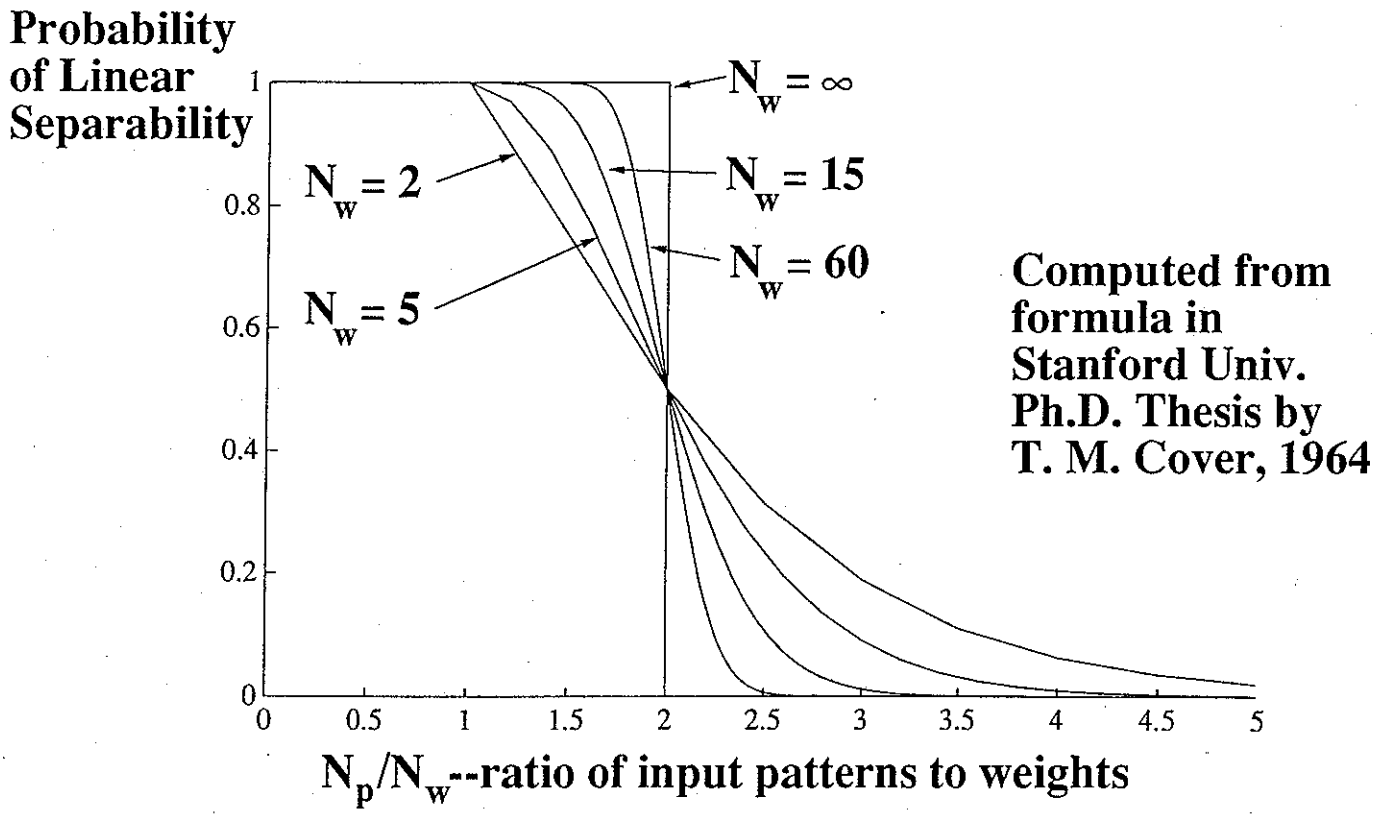


A two-Input Adaline

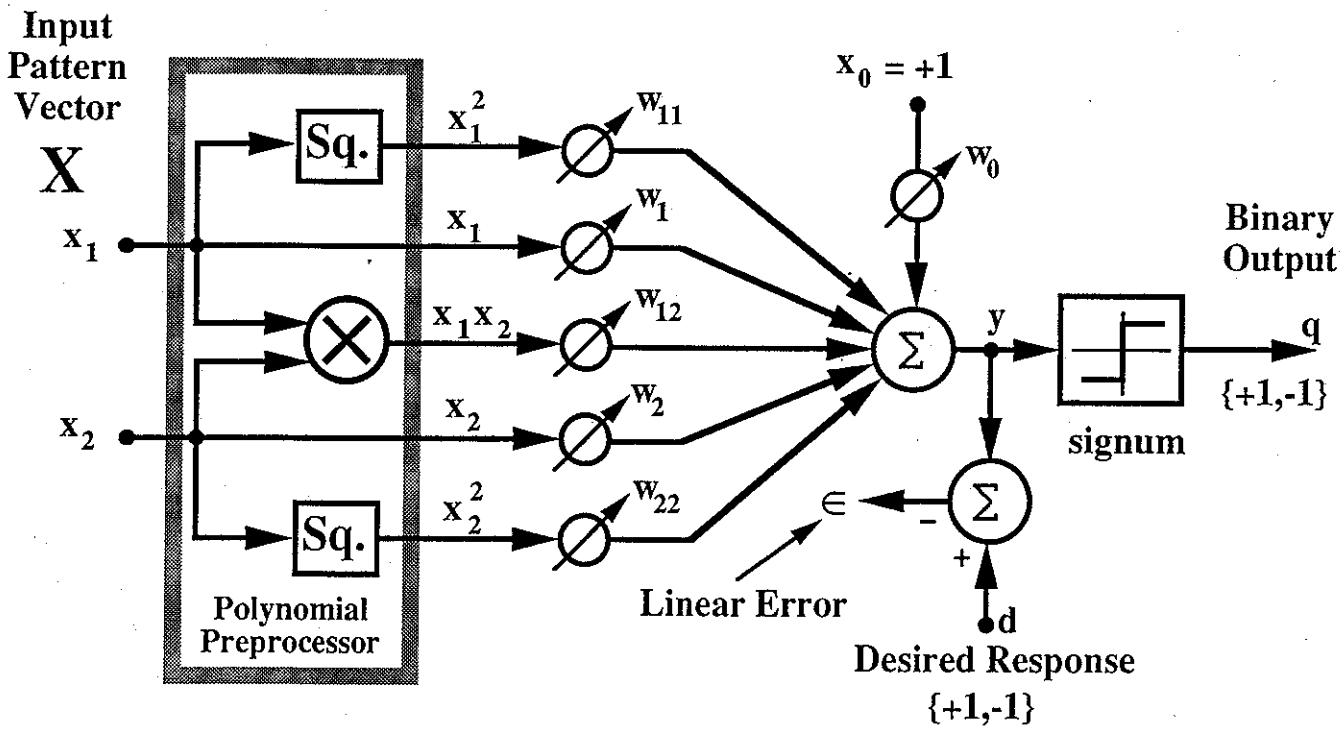


Separating line in pattern space

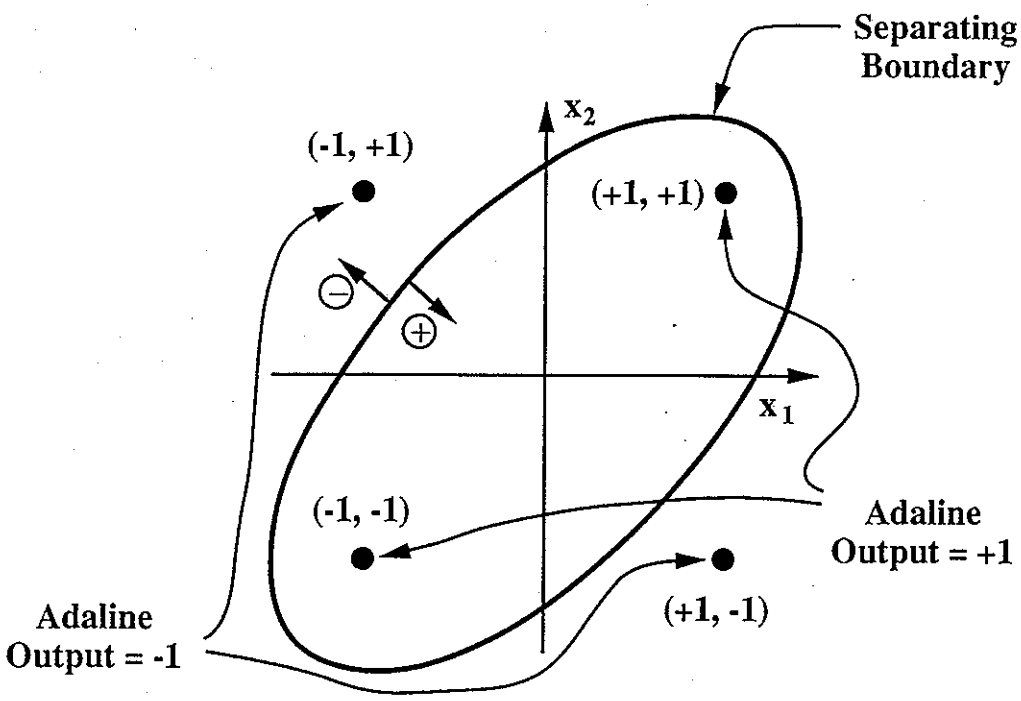
Adaline Capacity



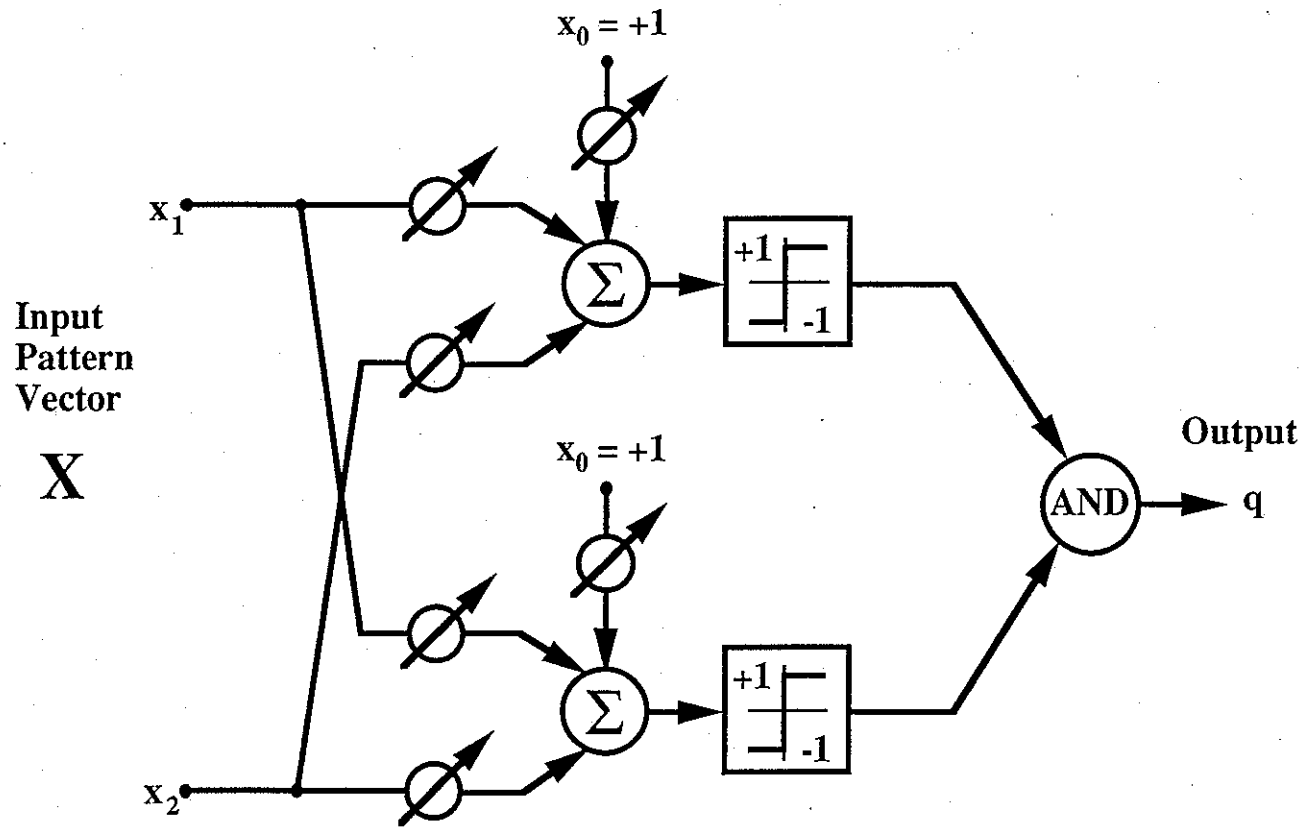
Computed from formula in Stanford Univ. Ph.D. Thesis by T. M. Cover, 1964



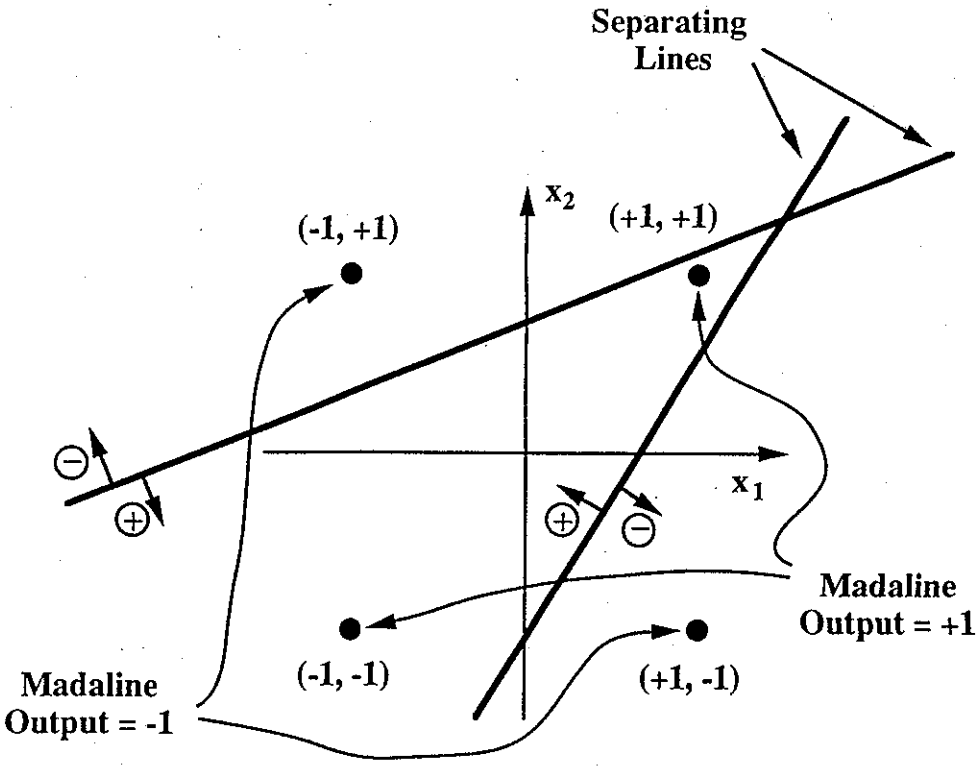
Adaline with polynomial preprocessor



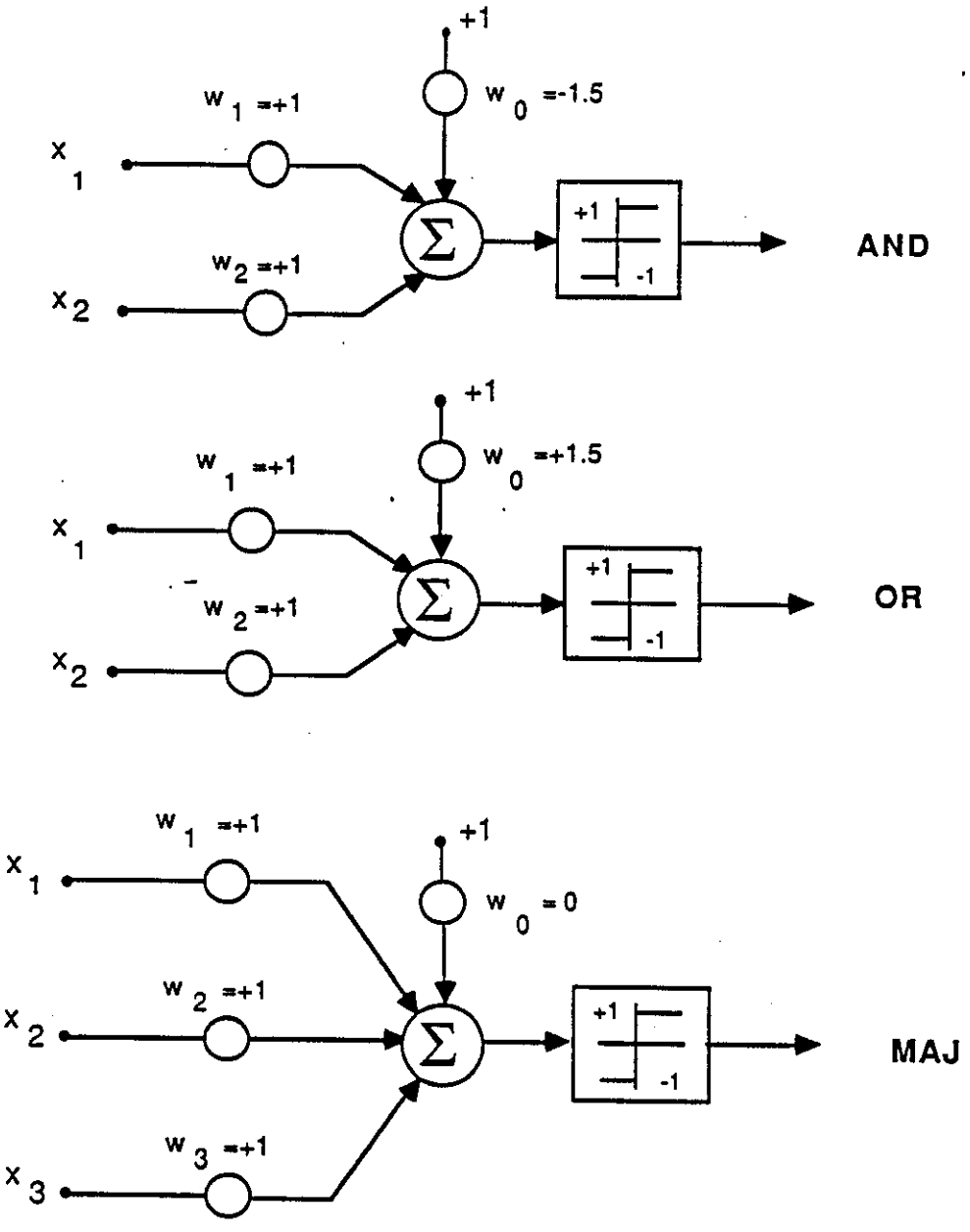
An elliptical separating boundary for the Exclusive NOR Function



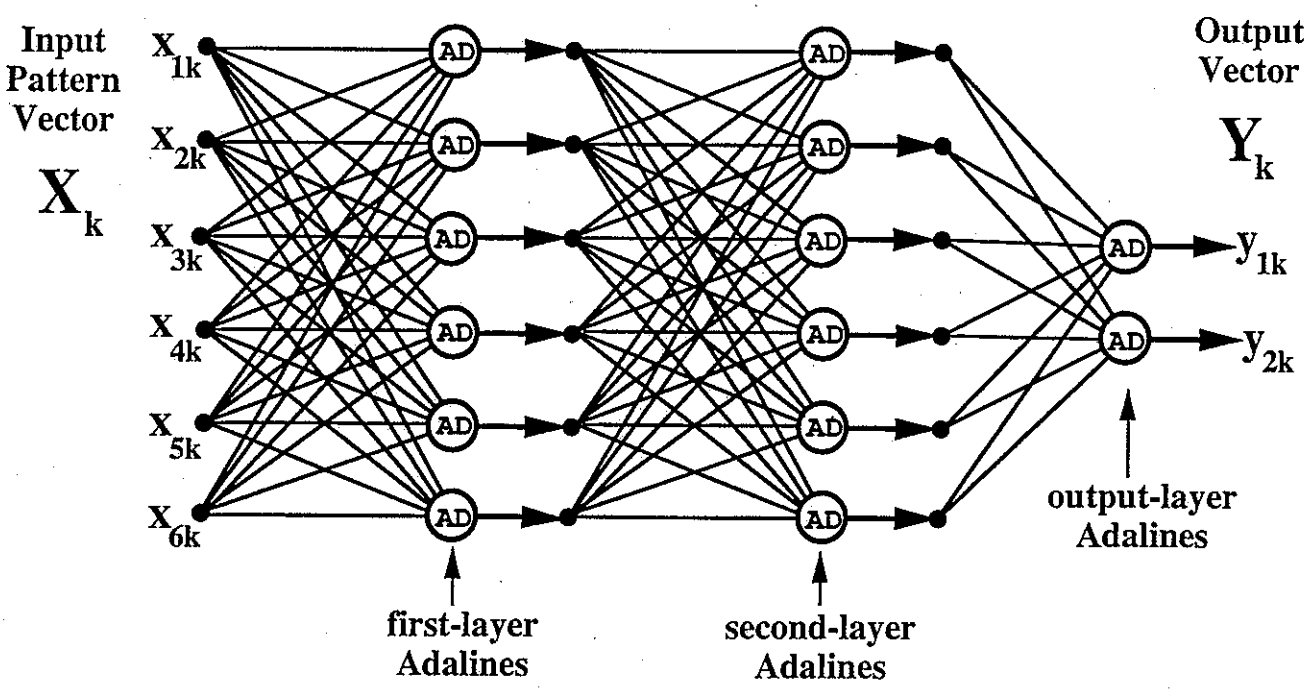
A two-Adaline form of Madaline I



Separating lines for the two-element Madaline



A neuronal implementation of AND, OR, and MAJ logic functions.



A three-layer adaptive neural network

Principle of Minimal Disturbance

Adapt to reduce the output error for the current training pattern with minimal disturbance to the responses already learned.

α-LMS Algorithm

$$\epsilon_k \triangleq d_k - \mathbf{w}_k^T \mathbf{x}_k. \tag{1}$$

Changing the weights yields a corresponding change in the error:

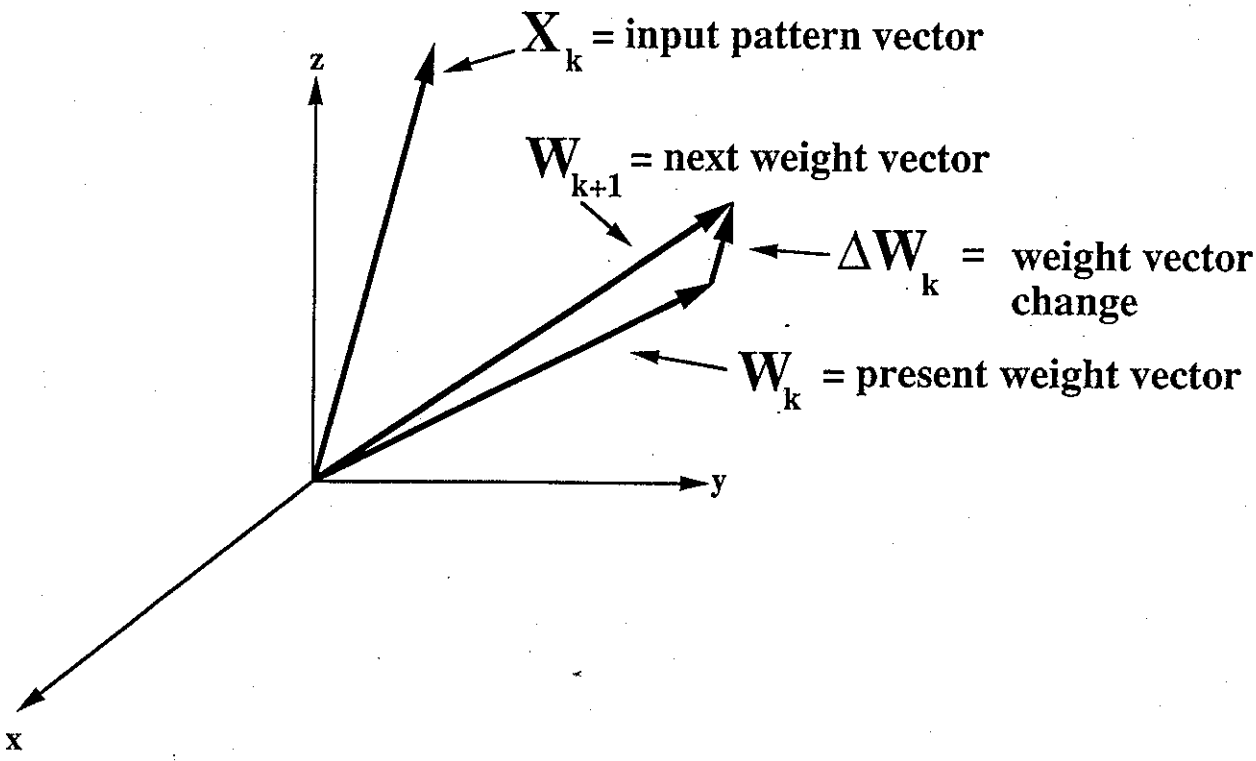
$$\Delta\epsilon_k = \Delta(d_k - \mathbf{w}_k^T \mathbf{x}_k) = -\mathbf{x}_k^T \Delta\mathbf{w}_k. \tag{2}$$

In accordance with the α-LMS rule, the weight change is as follows:

$$\Delta\mathbf{w}_k = \mathbf{w}_{k+1} - \mathbf{w}_k = \alpha \frac{\epsilon_k \mathbf{x}_k}{|\mathbf{x}_k|^2}. \tag{3}$$

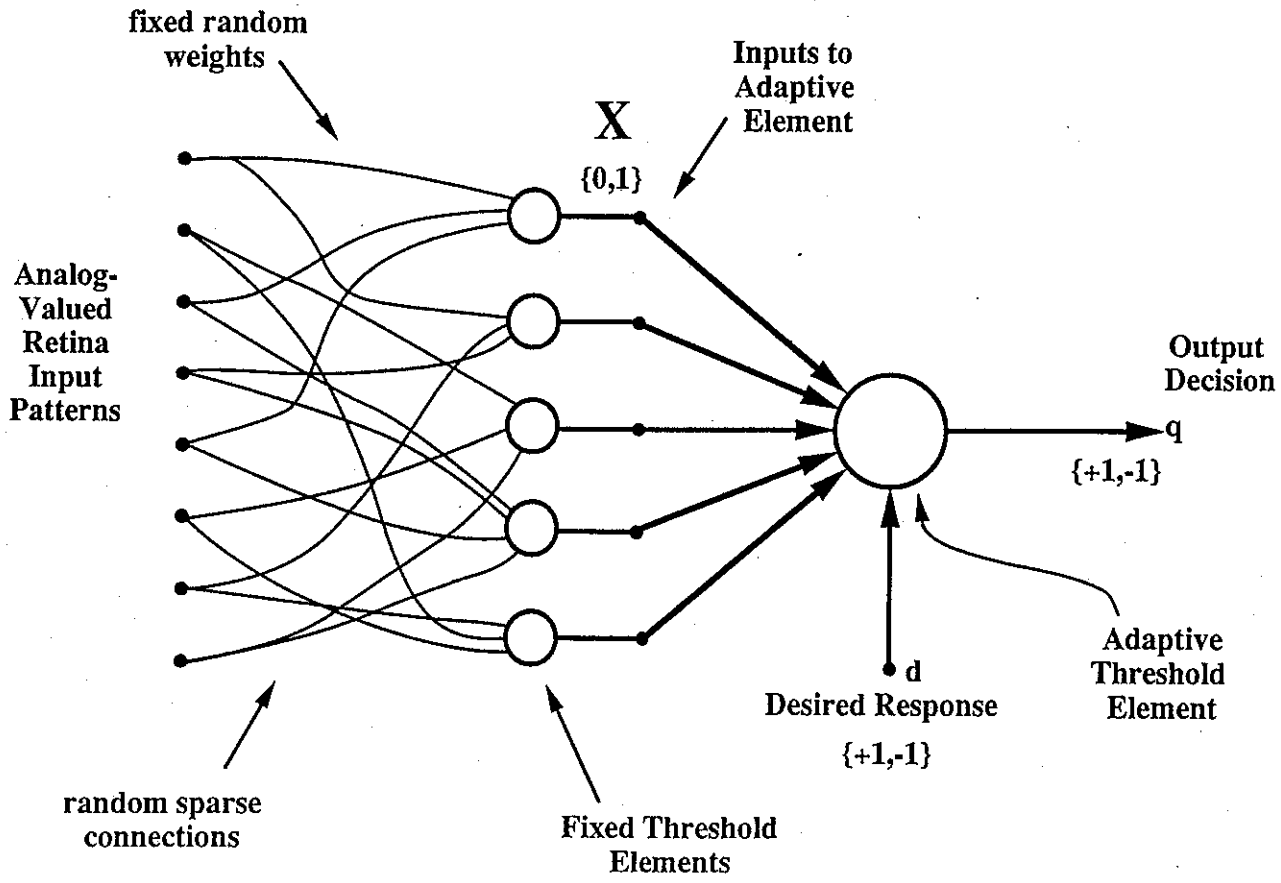
Combining Eqs. (2) and (3), we obtain

$$\Delta\epsilon_k = -\alpha \frac{\epsilon_k \mathbf{x}_k^T \mathbf{x}_k}{|\mathbf{x}_k|^2} = -\alpha\epsilon_k. \tag{4}$$

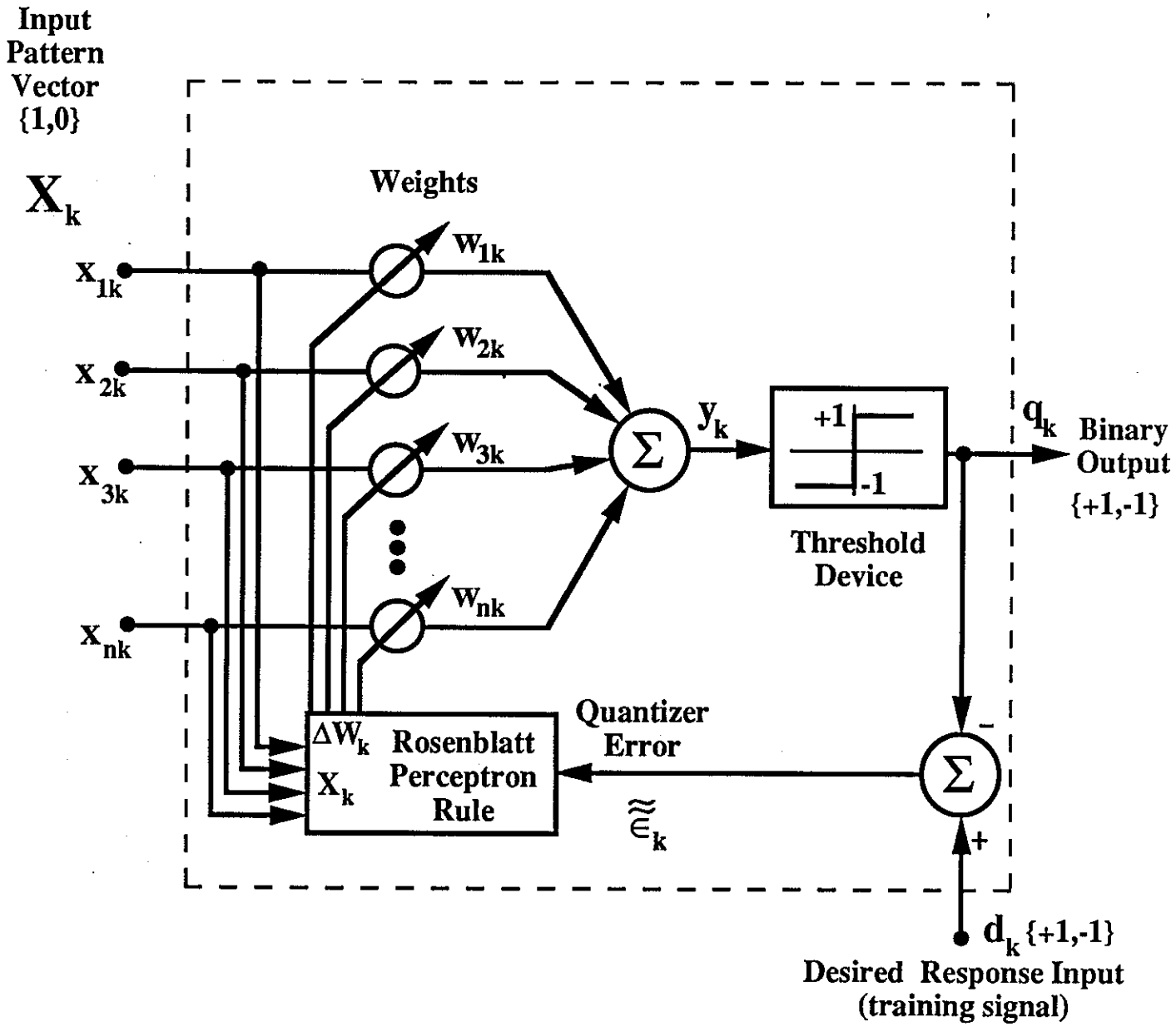


Weight correction by the LMS rule

Rosenblatt's Perceptron



Adaptive Threshold Element in the Perceptron



Perceptron Rule:

$$W_{k+1} = W_k + \mu \approx\approx\epsilon_k X_k$$

μ normally set to 1/2

Perceptron Rule

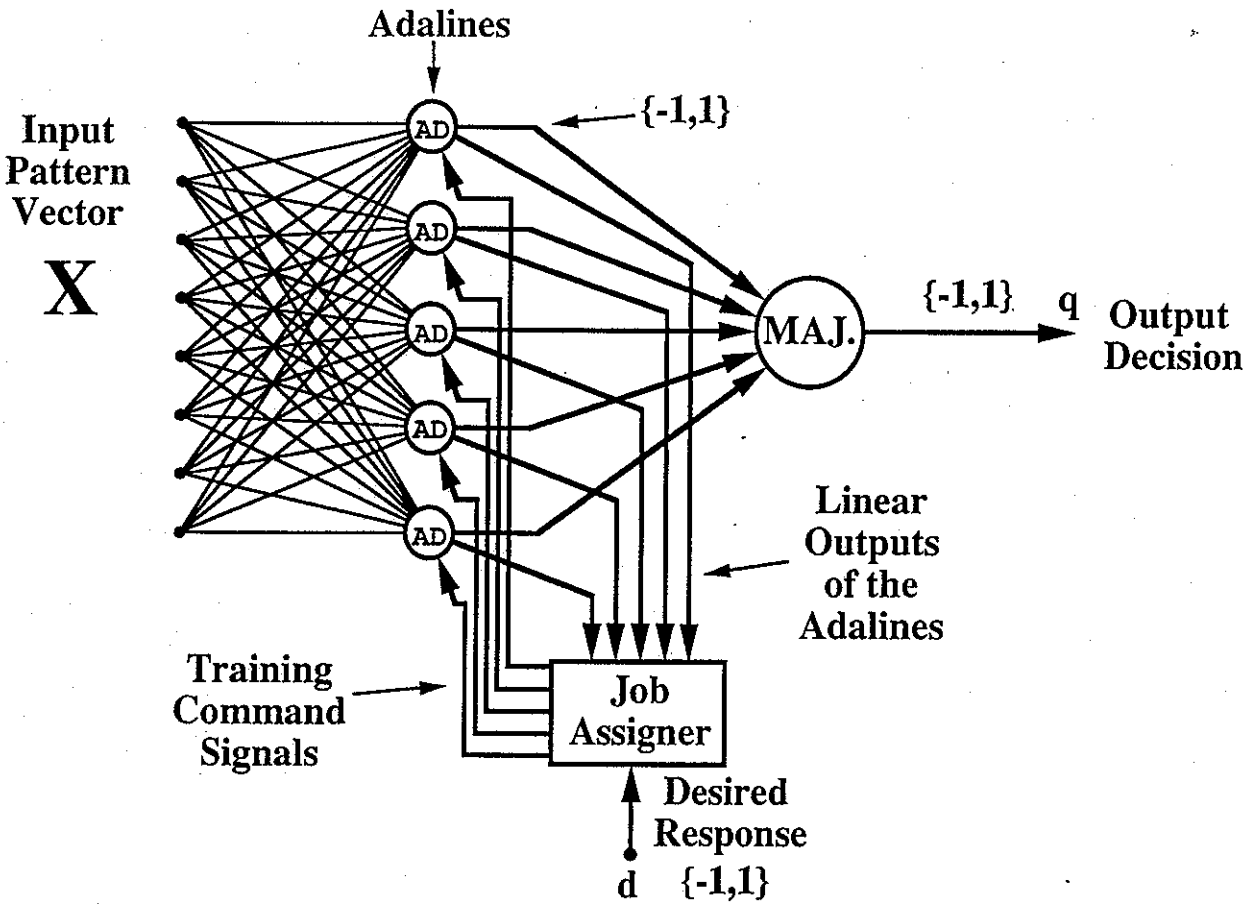
- If response is OK, do not adapt weights.
- Otherwise adapt weights by a fixed distance along the X-Vector to reduce error

Good Features

- Guaranteed to converge to solution if problem is linearly separable

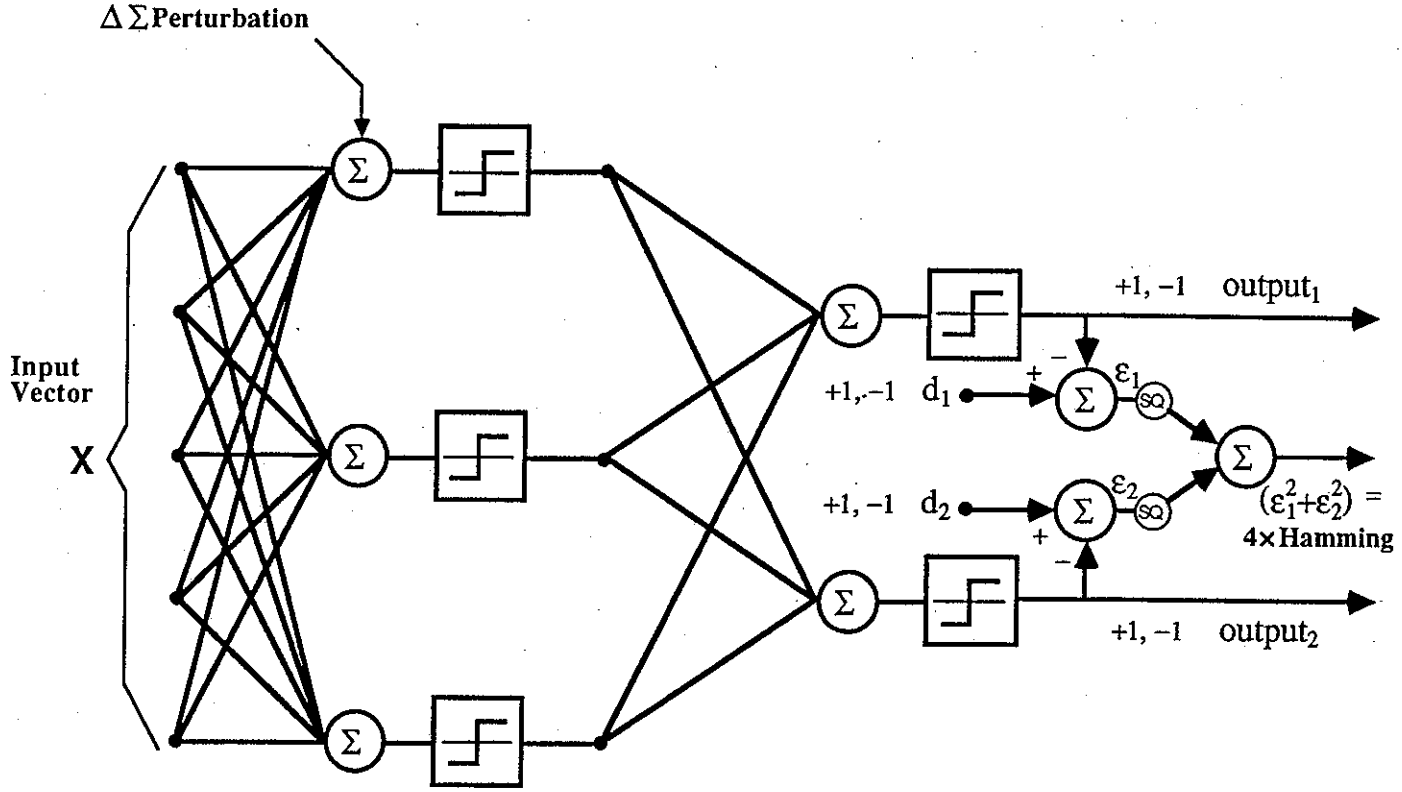
Bad Features

- Performs poorly if training set is not linearly separable.



A five-Adaline example of the MRI Architecture

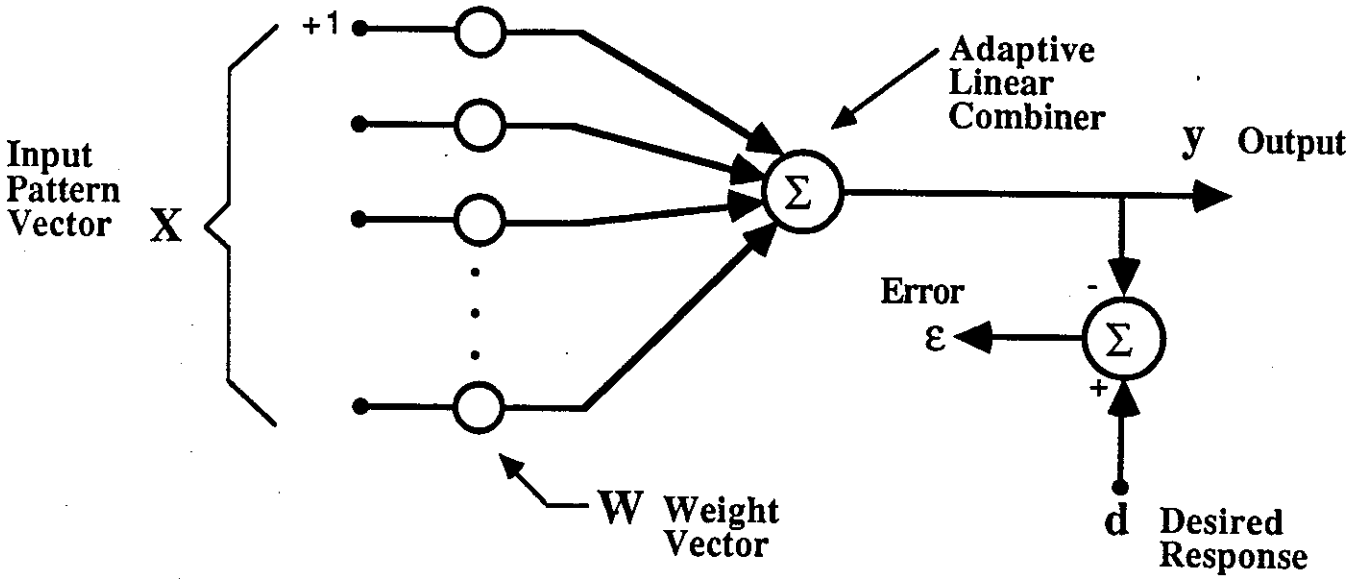
MRII of B. Widrow & R. Winter



For each layer, beginning with layer 1:

Toggle output of neuron with sum closest to zero. If output Hamming error is reduced, adapt neuron. Repeat for neuron whose sum is next closest to zero, etc. Can also adapt two at a time, etc. Adaptation reduces Hamming error.

Conventional LMS



Method of Steepest Descent $\rightarrow W_{k+1} = W_k + \mu(-\nabla_k)$

GRAD. $= \nabla = \dfrac{\partial E[\varepsilon^2]}{\partial W}$

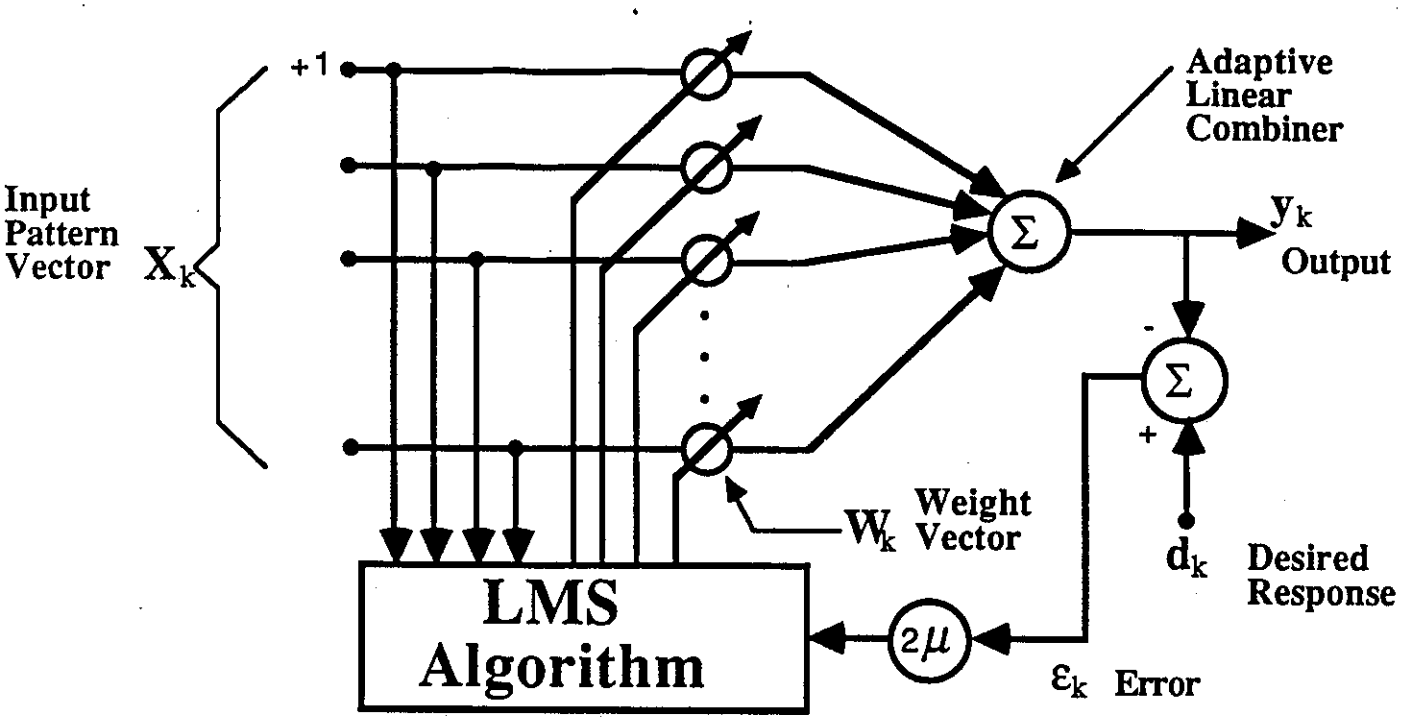
ERROR $= \varepsilon = d - X^T W$

INST. GRAD. $= \hat{\nabla} = \dfrac{\partial \varepsilon^2}{\partial W} = 2\varepsilon \dfrac{\partial \varepsilon}{\partial W} = -2\varepsilon X$

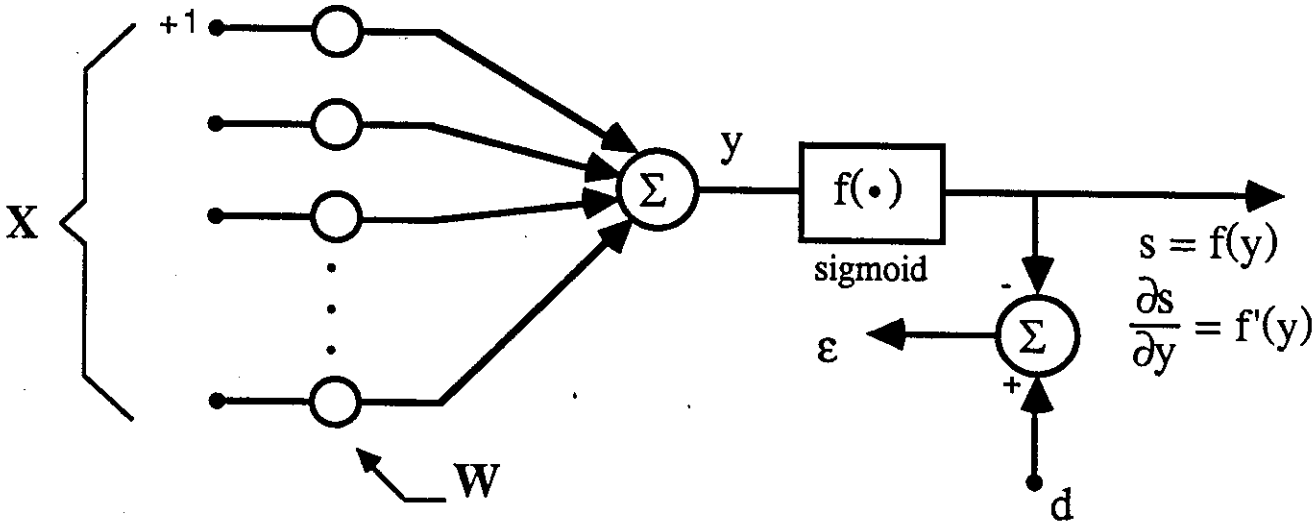
LMS $\rightarrow W_{k+1} = W_k + 2\mu\varepsilon_k X_k$

Implementation of Conventional LMS

$$\text{LMS} \rightarrow W_{k+1} = W_k + 2\mu\varepsilon_k X_k$$



"Sigmoid" LMS (BACK-PROP)



$$\textbf{GRAD.} = \nabla = \frac{\partial E[\varepsilon^2]}{\partial W}$$

$$\textbf{ERROR} = \varepsilon = d - f(X^T W)$$

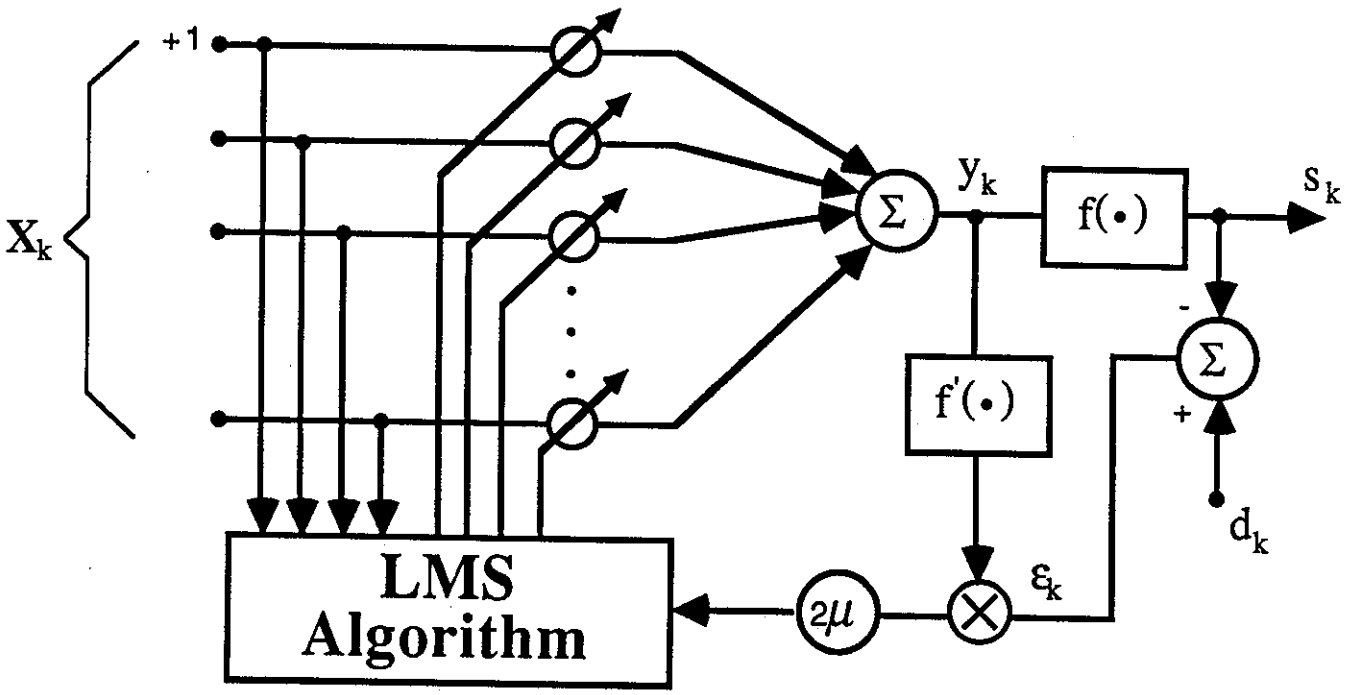
INST. GRAD. =

$$\hat{\nabla} = \frac{\partial \varepsilon^2}{\partial W} = 2\varepsilon \frac{\partial \varepsilon}{\partial W} = -2\varepsilon f'(X^T W) \frac{\partial (X^T W)}{\partial W} = -2\varepsilon X f'(X^T W)$$

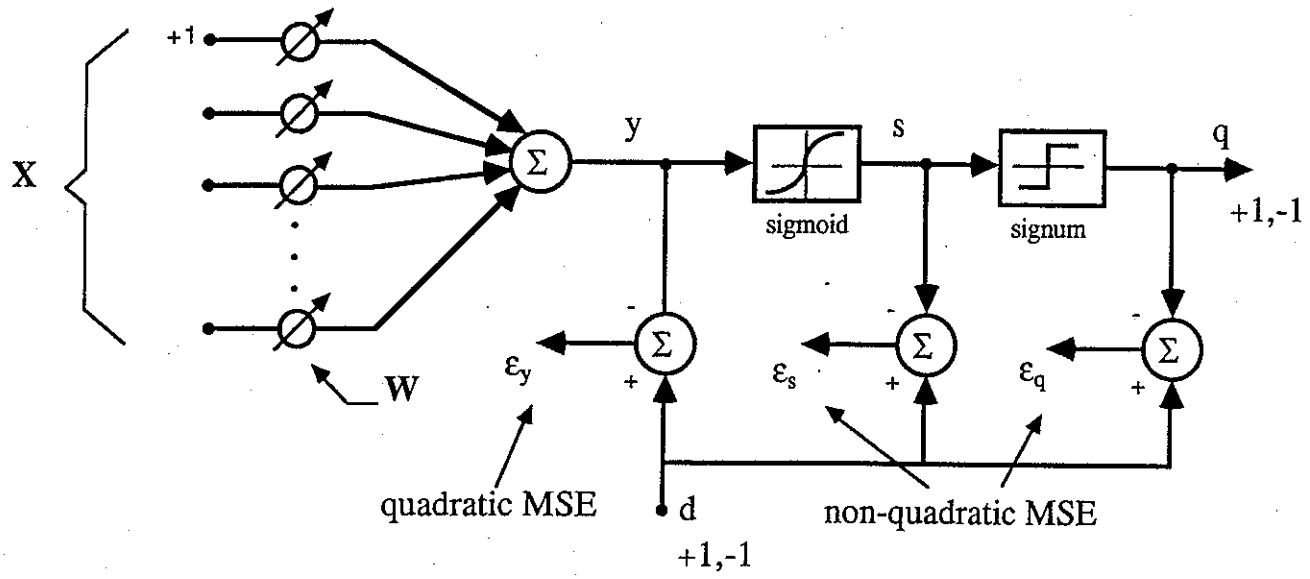
SIGMOID LMS (BACK-PROP) $\rightarrow W_{k+1} = W_k + 2\mu\varepsilon_k X_k f'(X_k^T W_k)$

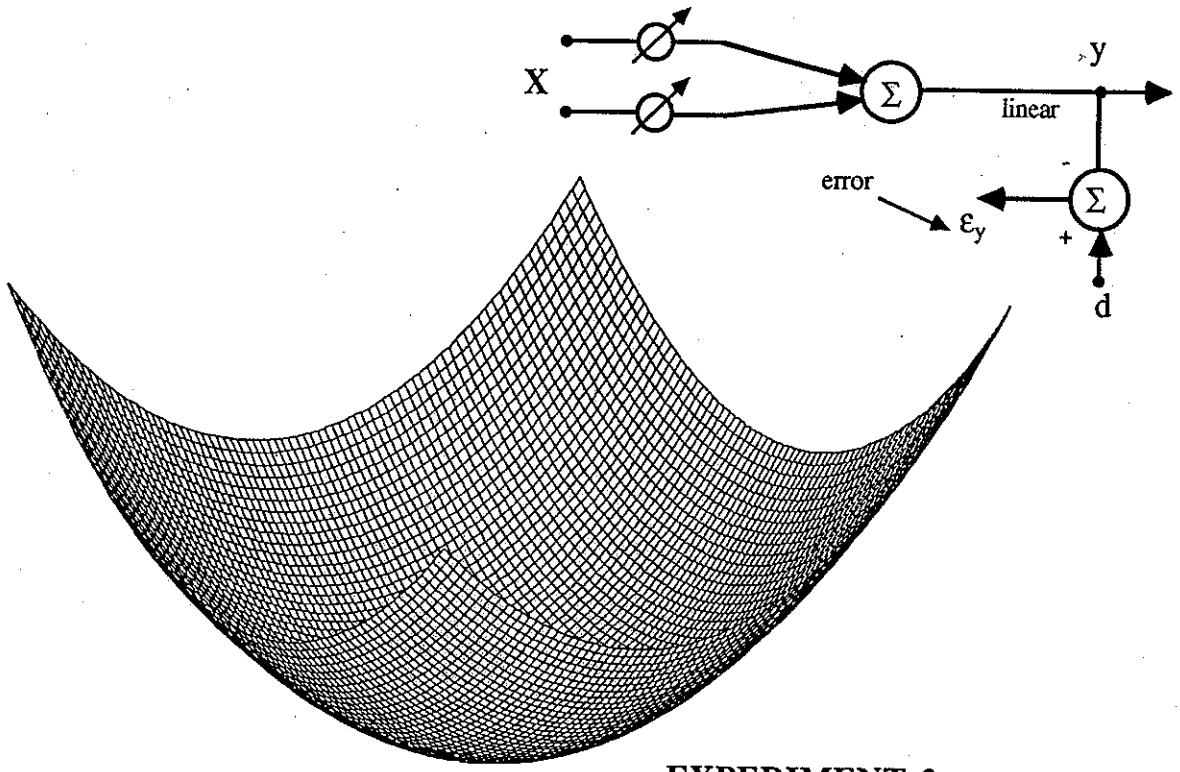
Implementation of "Sigmoid" LMS (BACK-PROP)

$$\text{SIGMOID LMS} \rightarrow W_{k+1} = W_k + 2\mu\varepsilon_k X_k f'(X_k^T W_k)$$



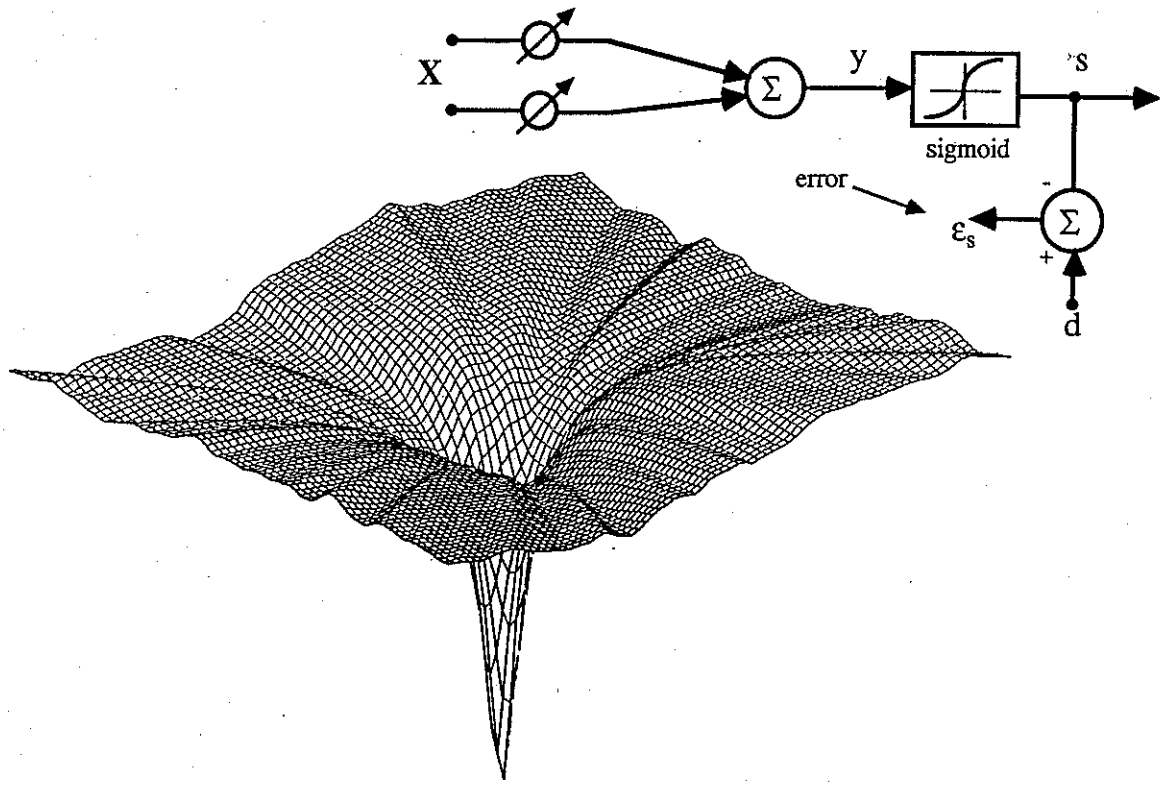
LINEAR MSE and Sigmoid MSE and Signum MSE





EXPERIMENT 2a
Same input as EXPT. 1a,
but different desired response.

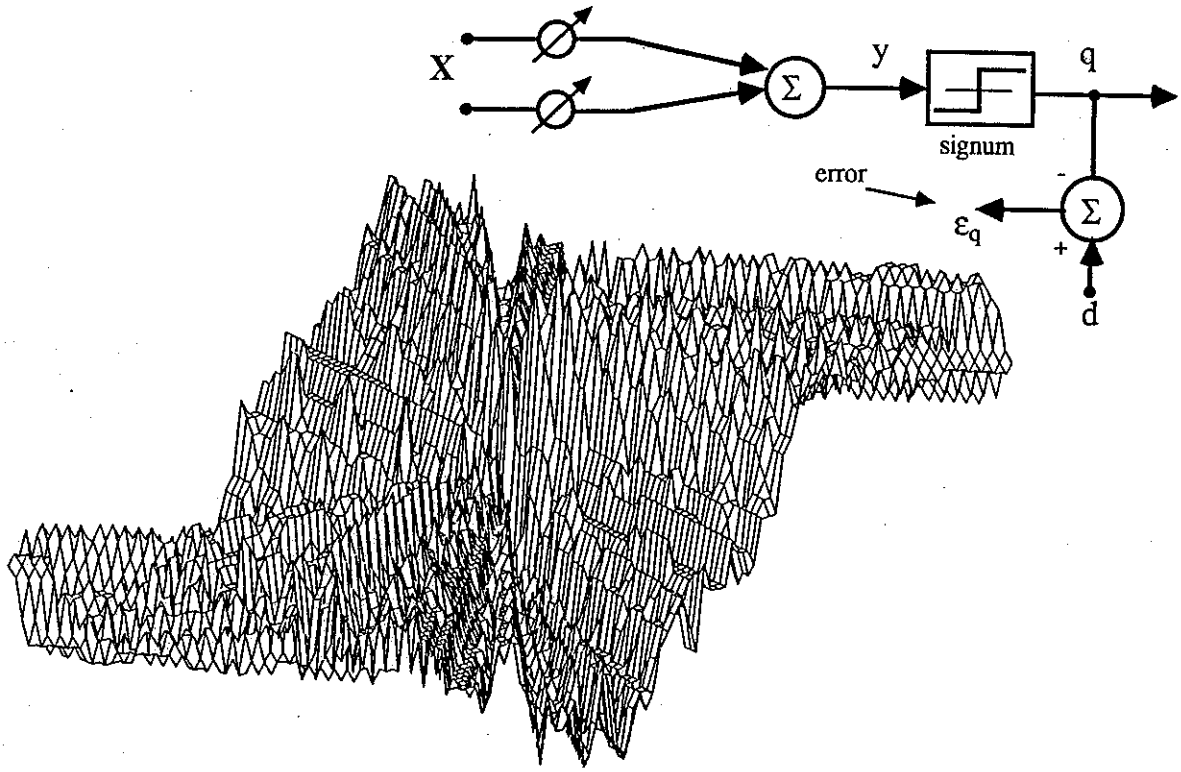
Mean Squared Error Surface (linear)



EXPERIMENT 2b

Same input as EXPT. 1b, but different desired response.

Mean Squared Error Surface (sigmoid)

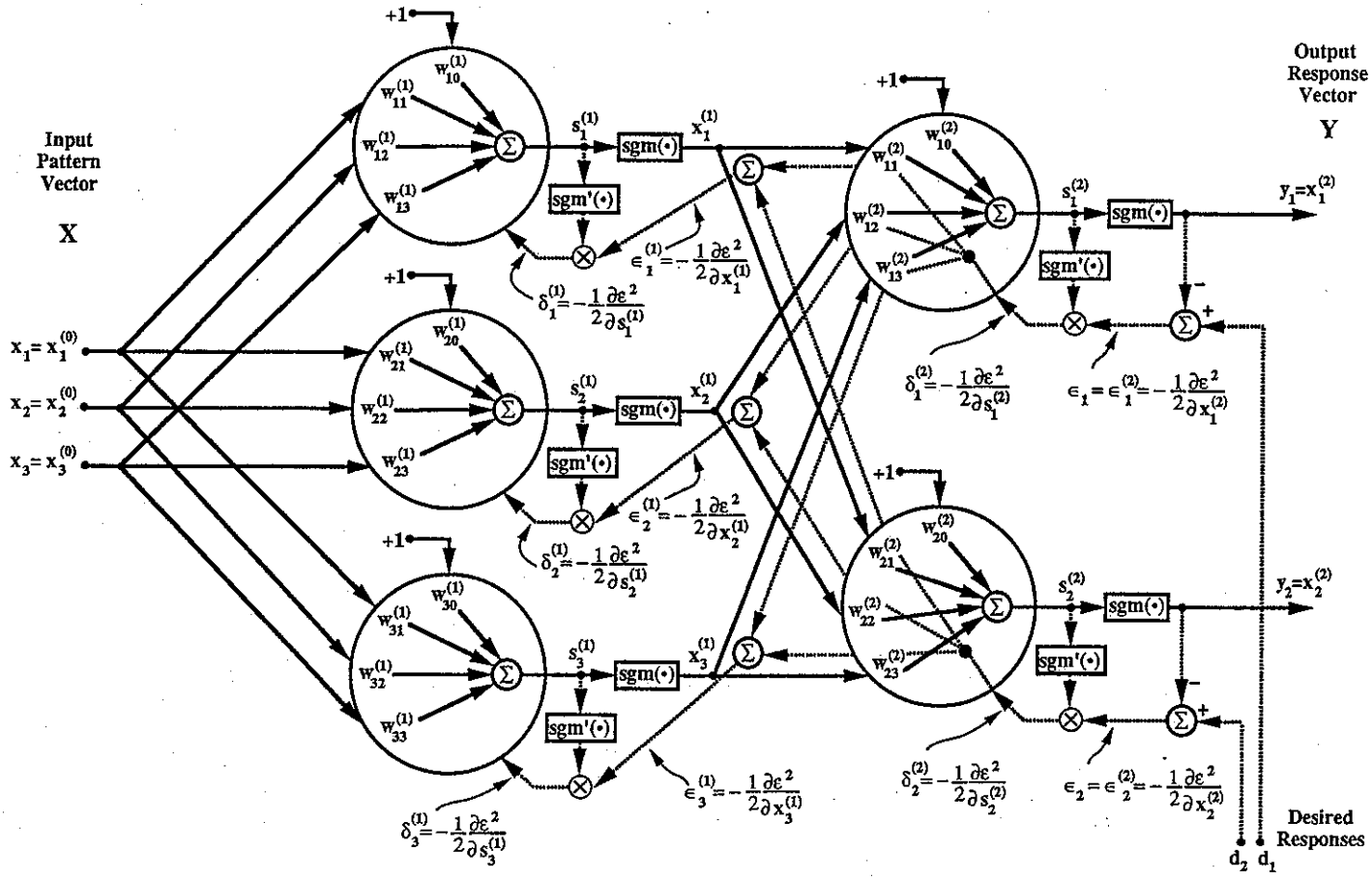


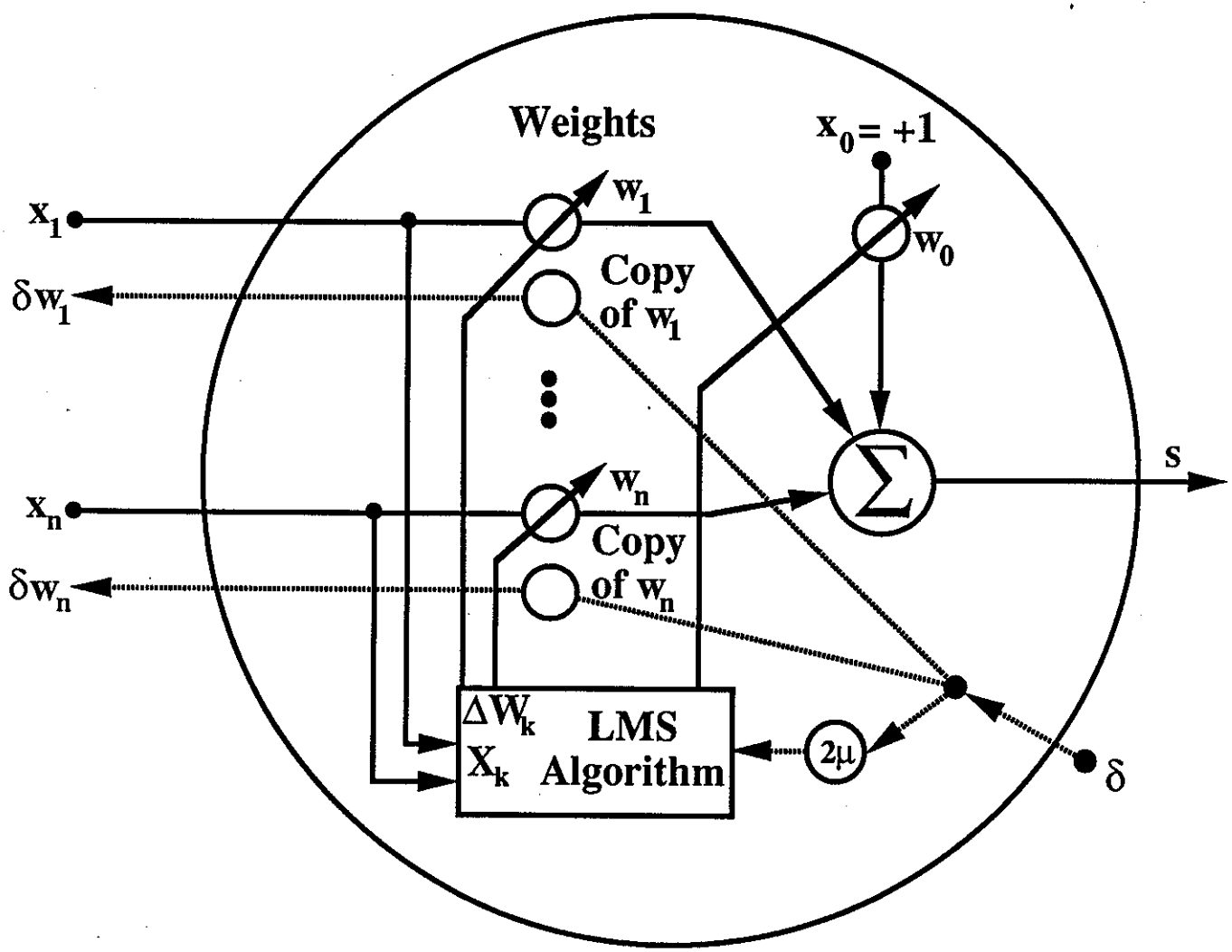
EXPERIMENT 2c

Same input as EXPT. 1c,
but different desired response.

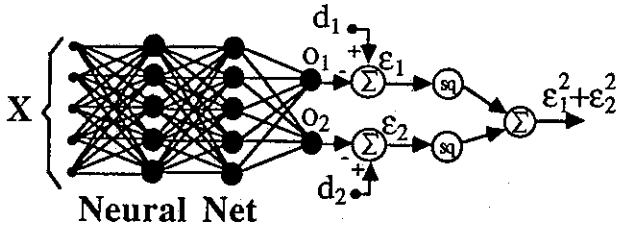
Mean Squared Error Surface (signum)

Backpropagation Network

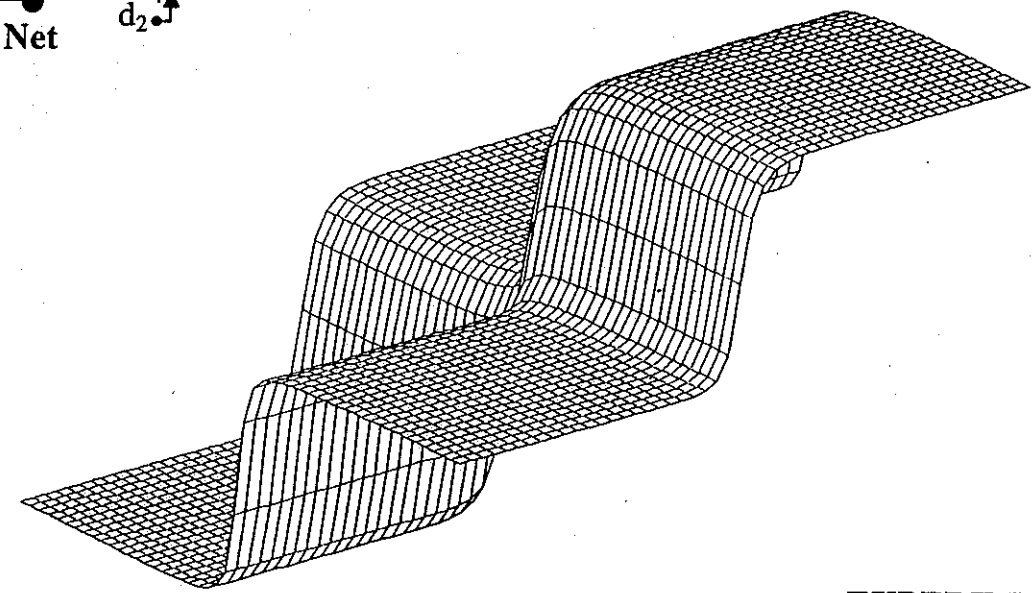




Detail of Backpropagation Node

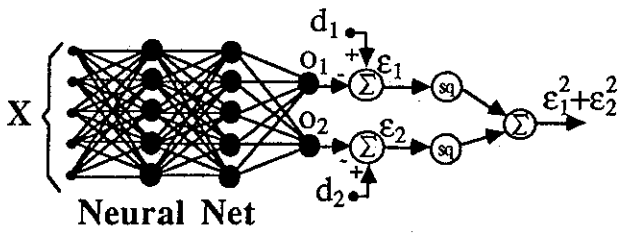


- 4 inputs
- 3-layer network: 5 feed 8 feed 2
- sigmoids

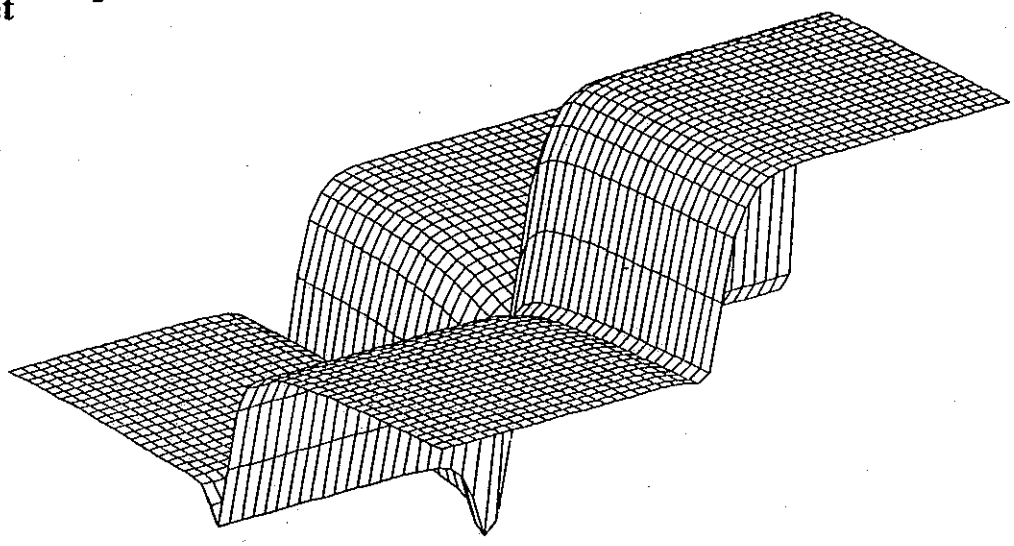


EXPERIMENT 3a.
Randomize weights, then vary two first-layer weights.

Mean Square Error Surface

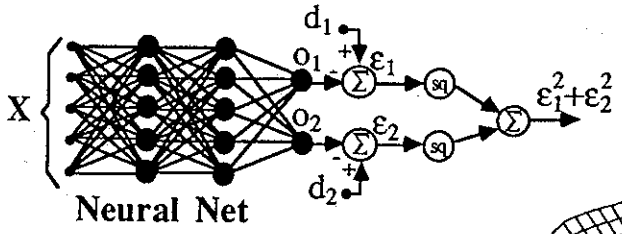


- 4 inputs
- 3-layer network: 5 feed 8 feed 2
- sigmoids

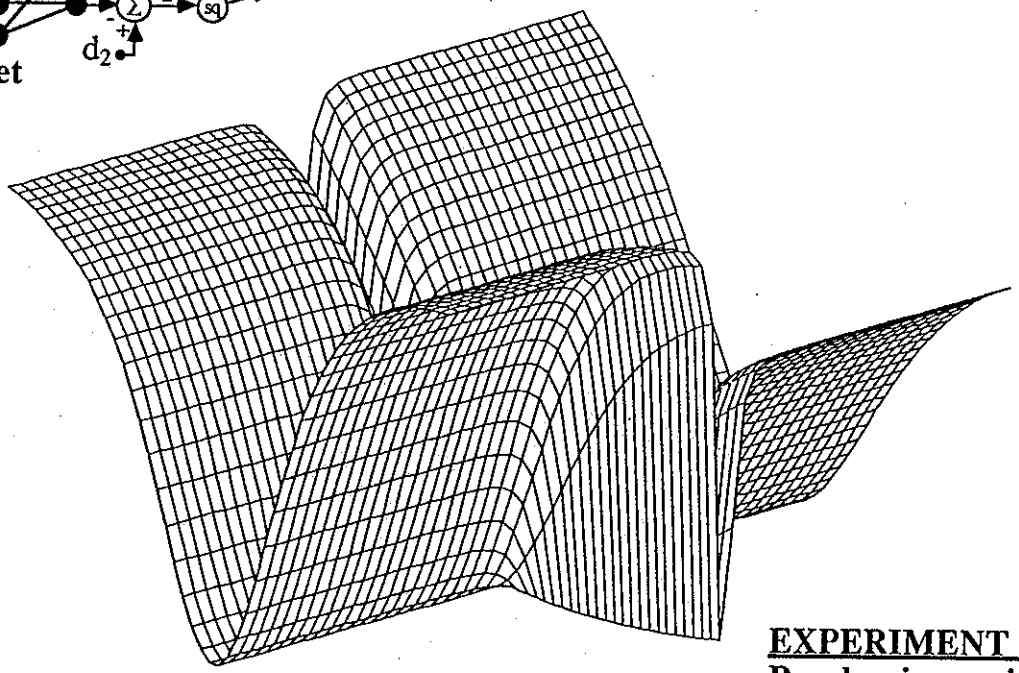


EXPERIMENT 3b.
Adapt weights by backprop, then vary two first-layer weights.

Mean Square Error Surface

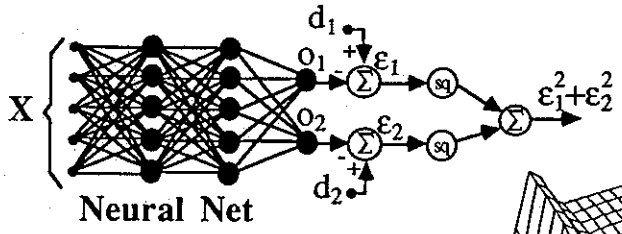


- 4 inputs
- 3-layer network: 5 feed 8 feed 2
- sigmoids

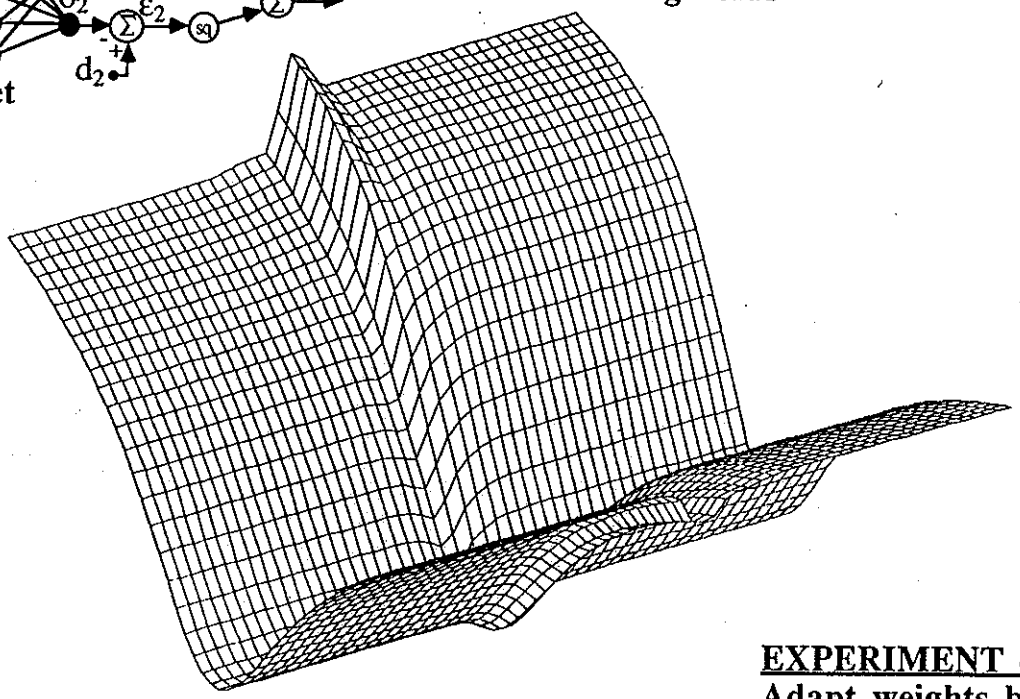


<u>EXPERIMENT 4a.</u>
Randomize weights, then vary one first-layer weight and one third-layer weight.

Mean Square Error Surface



- 4 inputs
- 3-layer network: 5 feed 8 feed 2
- sigmoids



EXPERIMENT 4b.
Adapt weights by backprop, then vary one first-layer weight and one third-layer weight.

Mean Square Error Surface

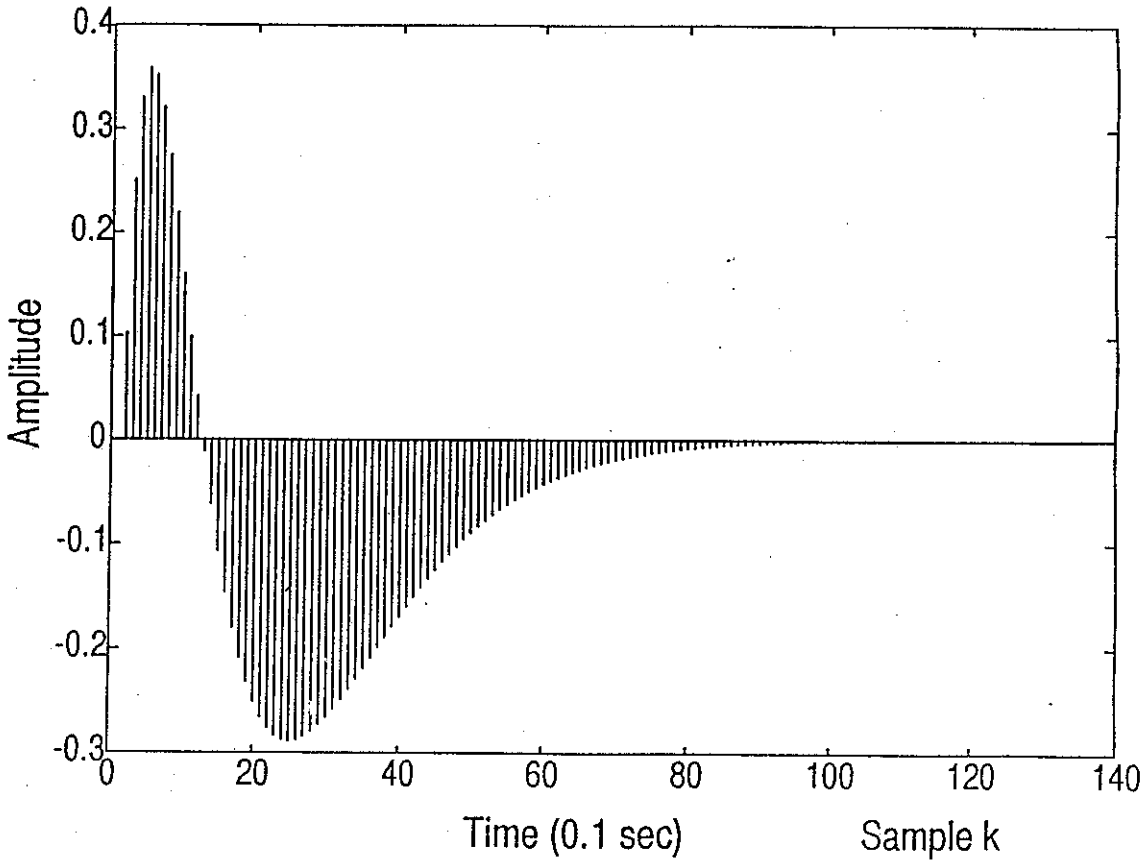
MICHEL BILELLO

AND

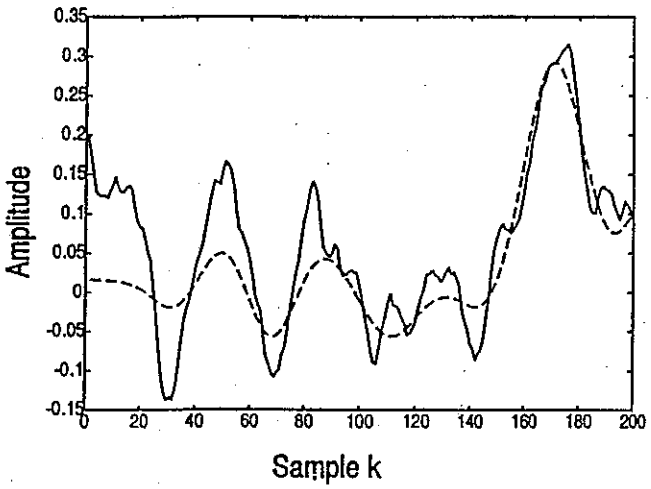
B. WIDROW

LINEAR AND NONLINEAR ADAPTIVE INVERSE CONTROL

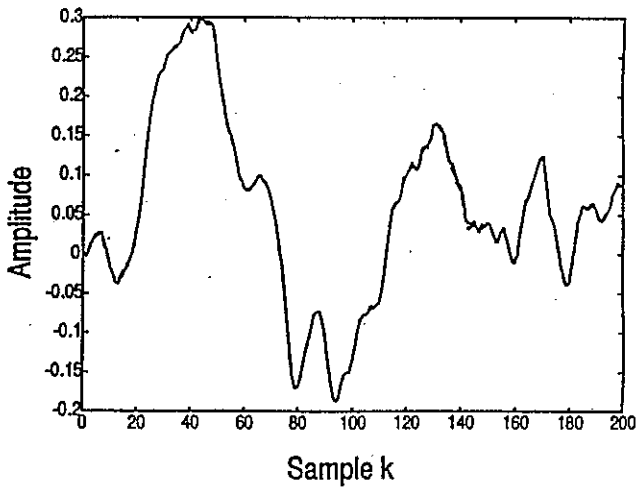
Impulse response of non-minimum phase plant



Dynamic tracking performance of non-minimum phase plant



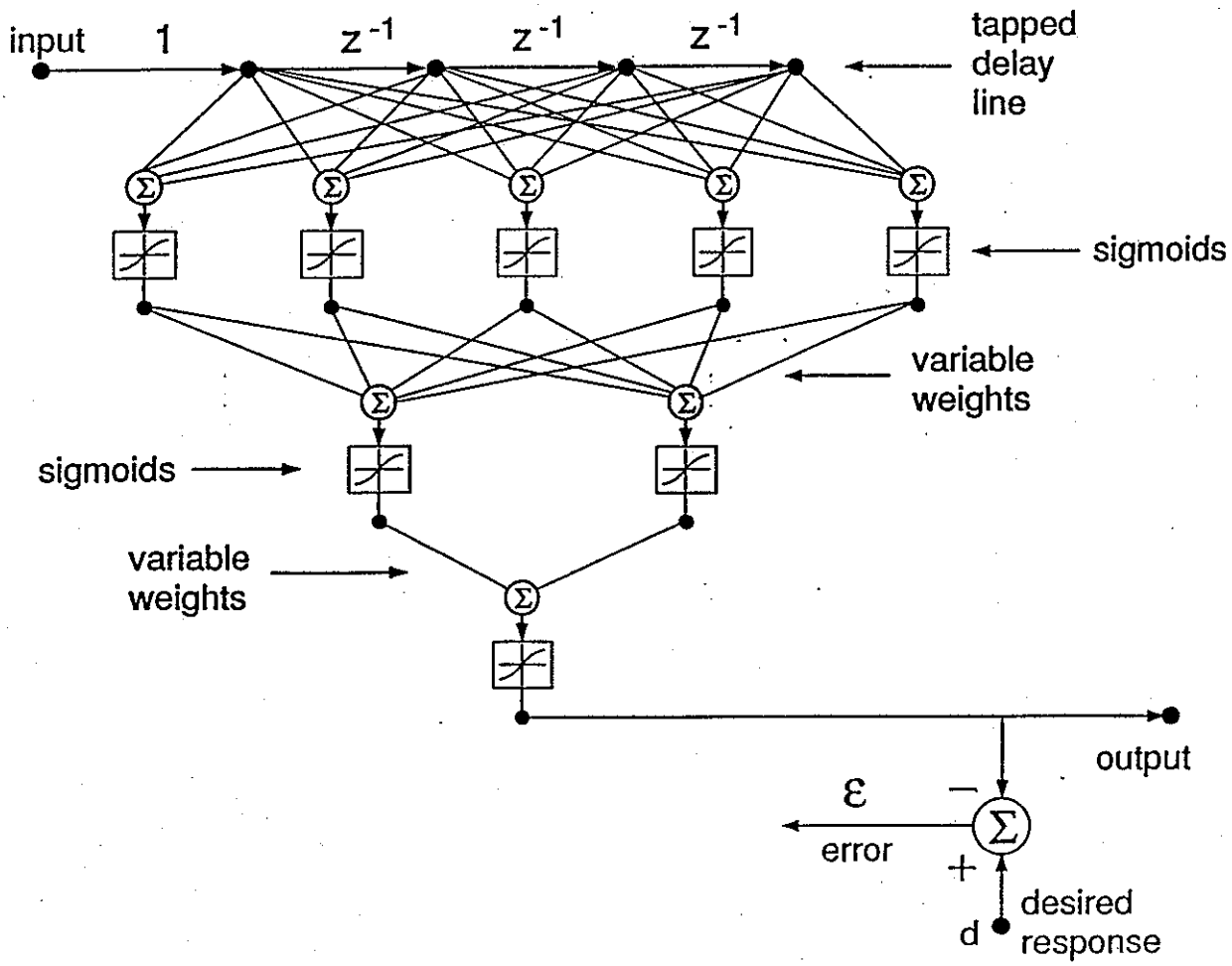
Beginning of training

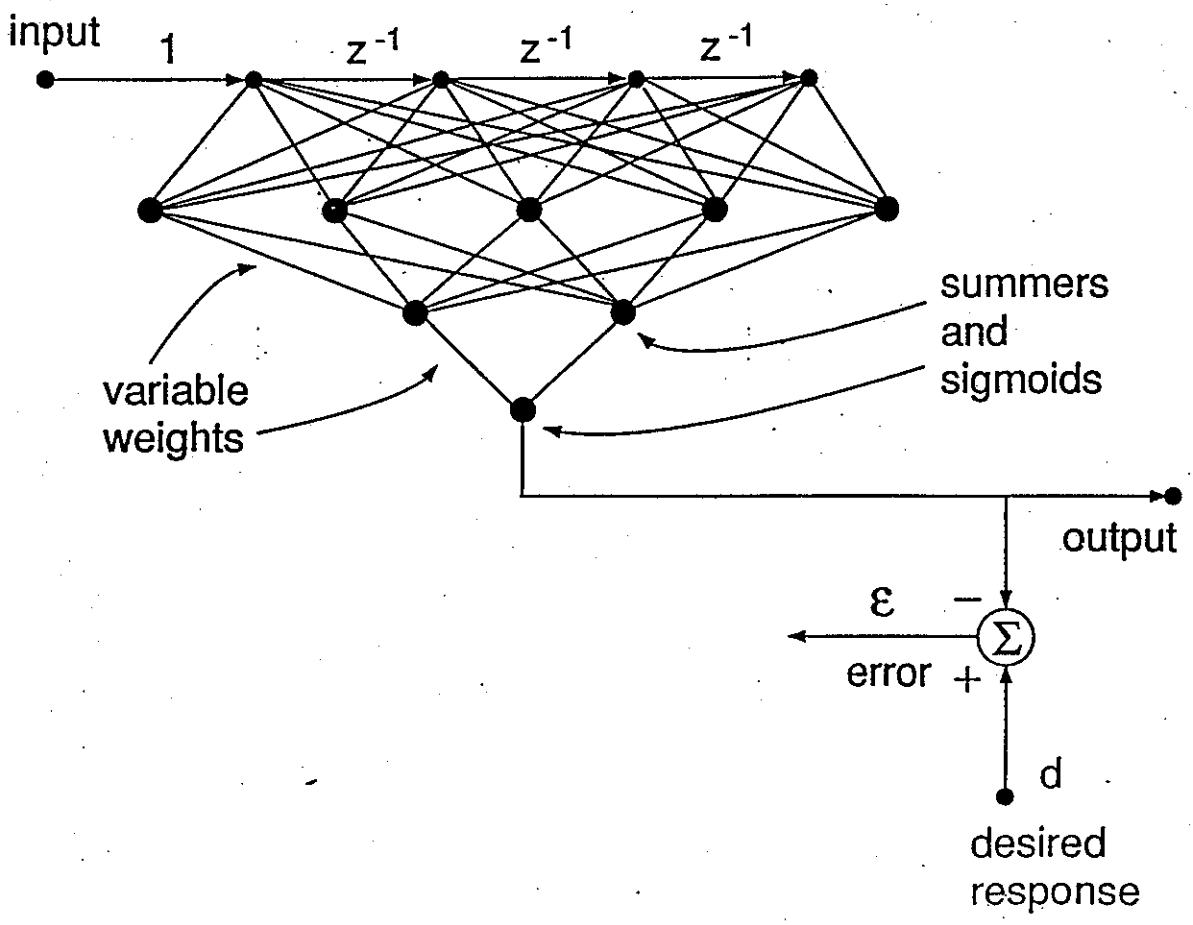


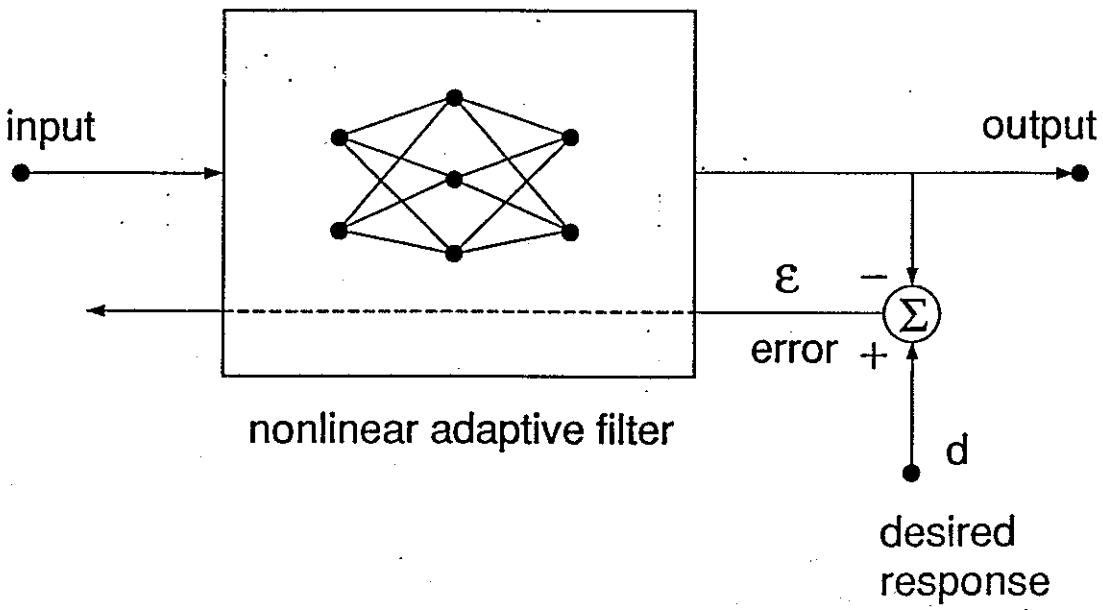
End of training

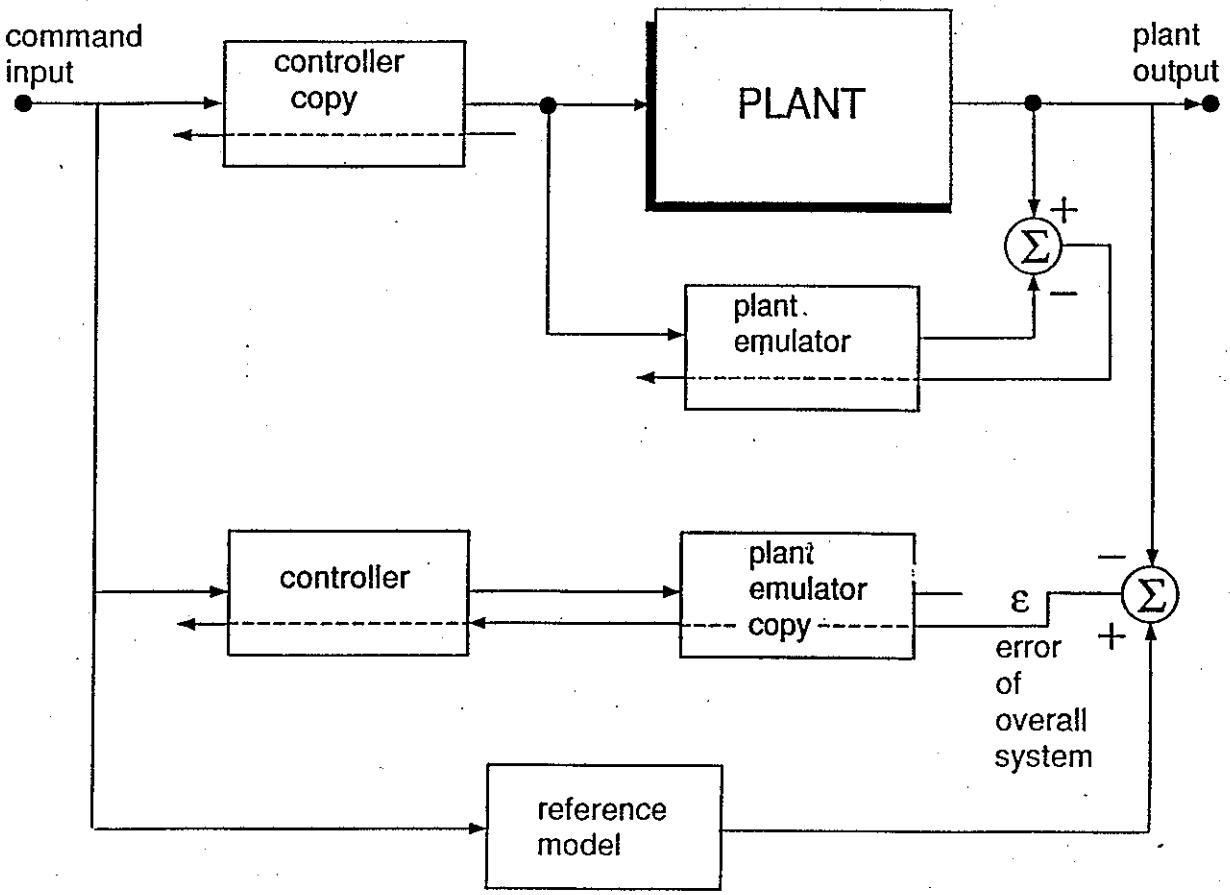
——— Reference model output (including delay)

------ Plant output

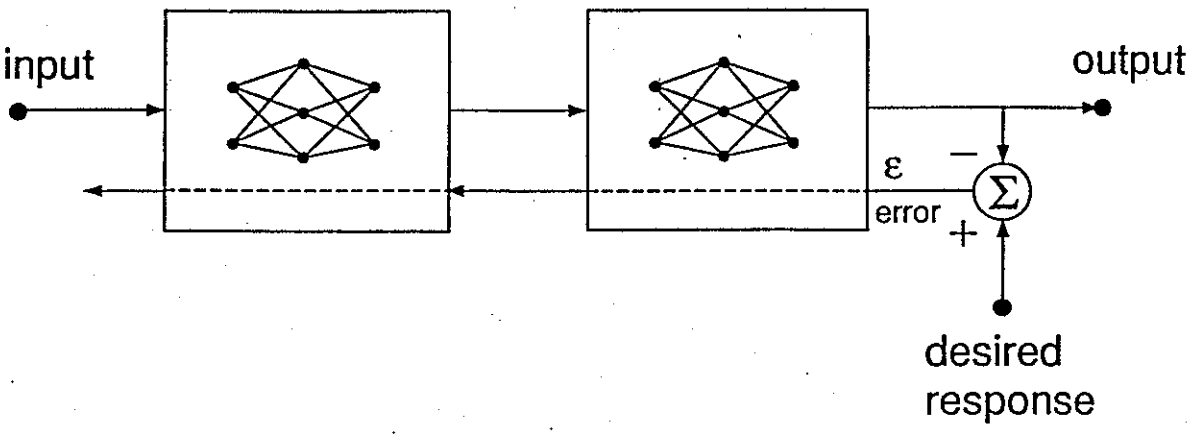

input
1
z^{-1}
z^{-1}
z^{-1}
tapped
delay
line
sigmoids
variable
weights
sigmoids
variable
weights
output
ε
error
−
+
desired
response
d

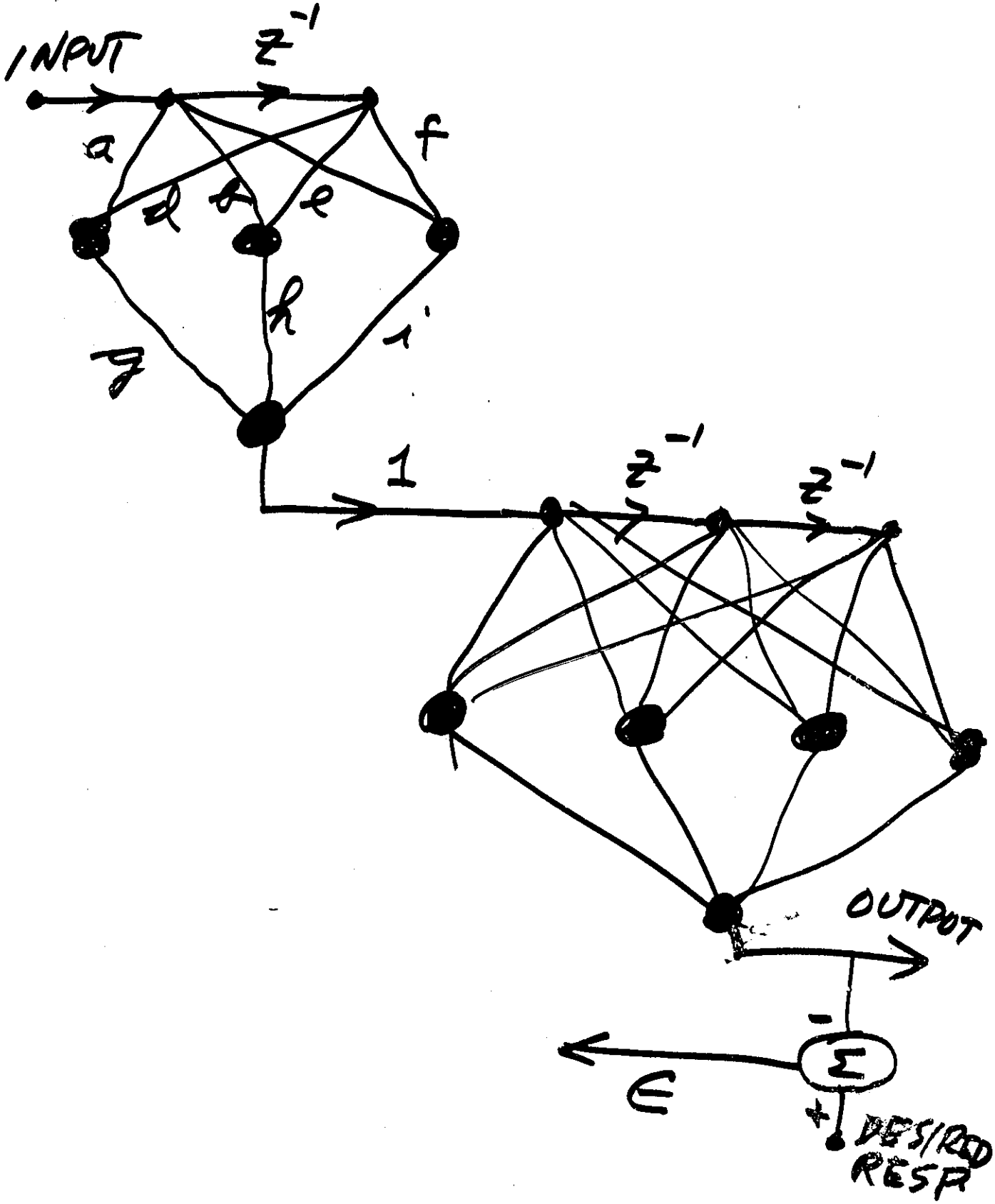

input
1
z^{-1}
z^{-1}
z^{-1}
summers
and
sigmoids
variable
weights
output
ε
−
error
+
Σ
d
desired
response

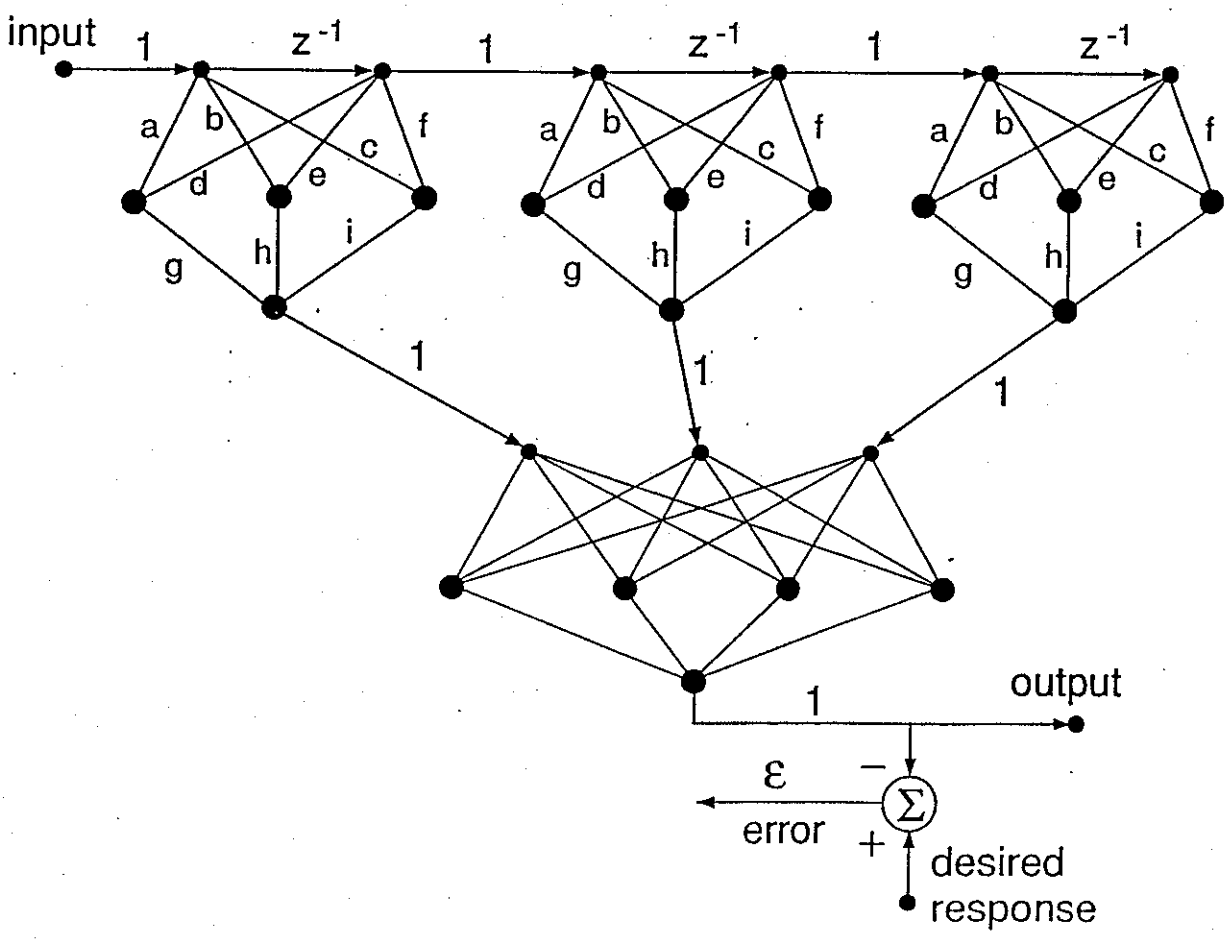

input
output
ε
−
Σ
error
+
nonlinear adaptive filter
d
desired
response


command input
controller copy
PLANT
plant output
+
Σ
−
plant emulator
controller
plant emulator copy
−
ε
Σ
+
error of overall system
reference model

(9)

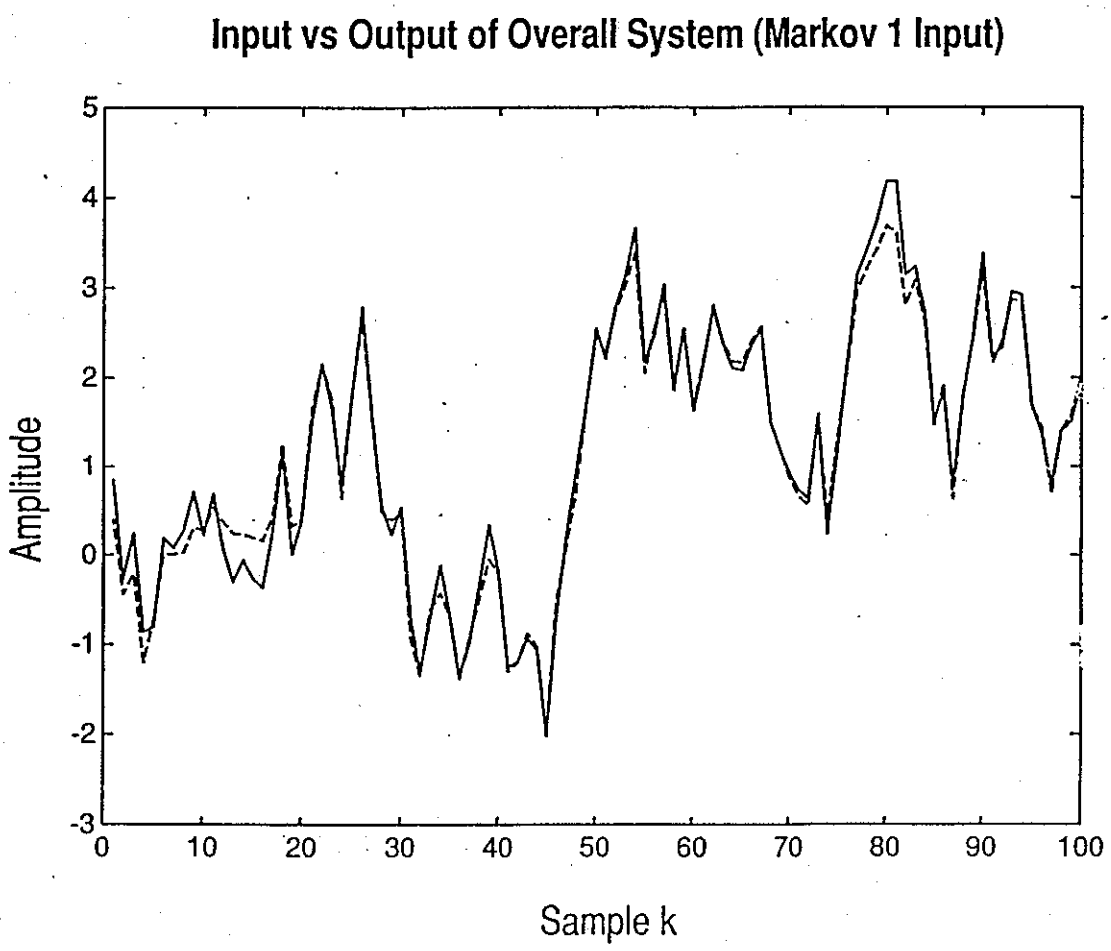

input
output
ε
error
−
+
Σ
desired
response

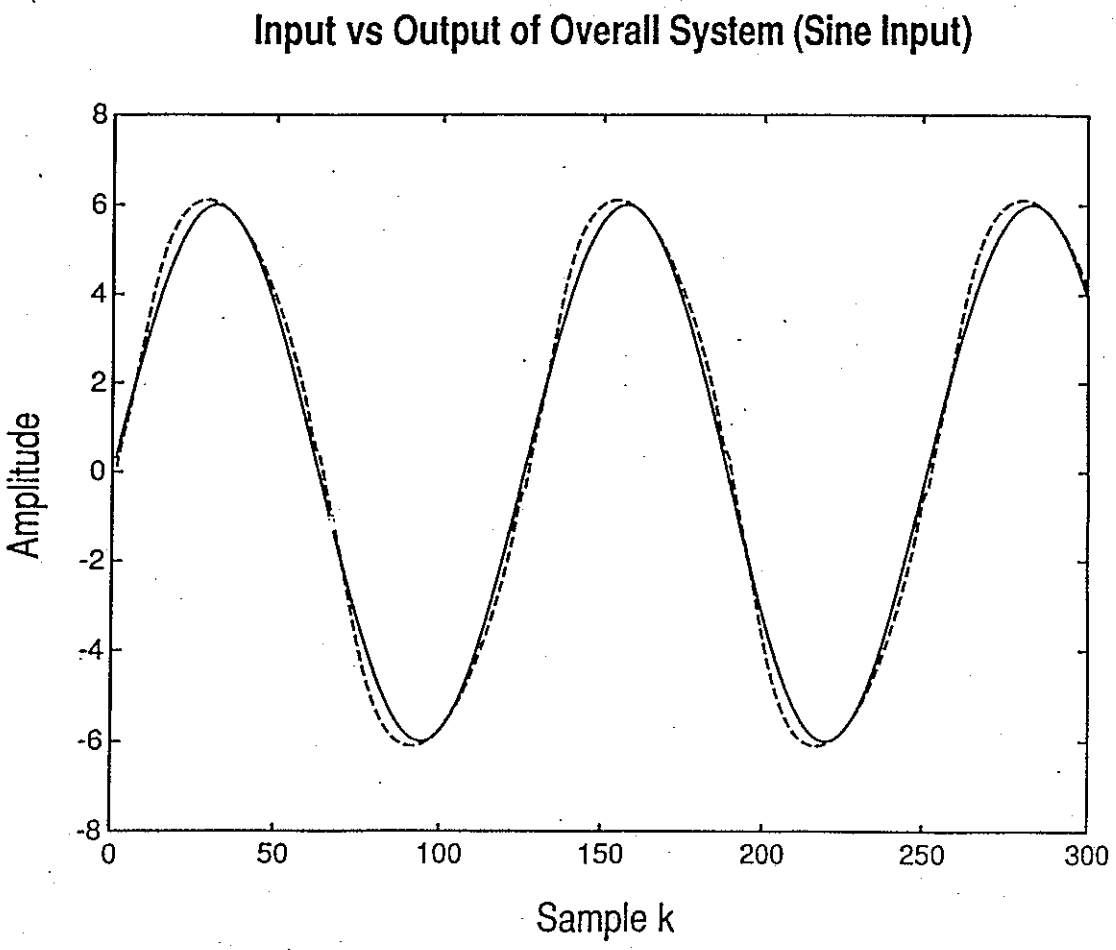

INPUT
z^{-1}
a
f
d
h
e
k
g
1
z^{-1}
z^{-1}
OUTPUT
-
Σ
+
ϵ
DESIRED
RESP

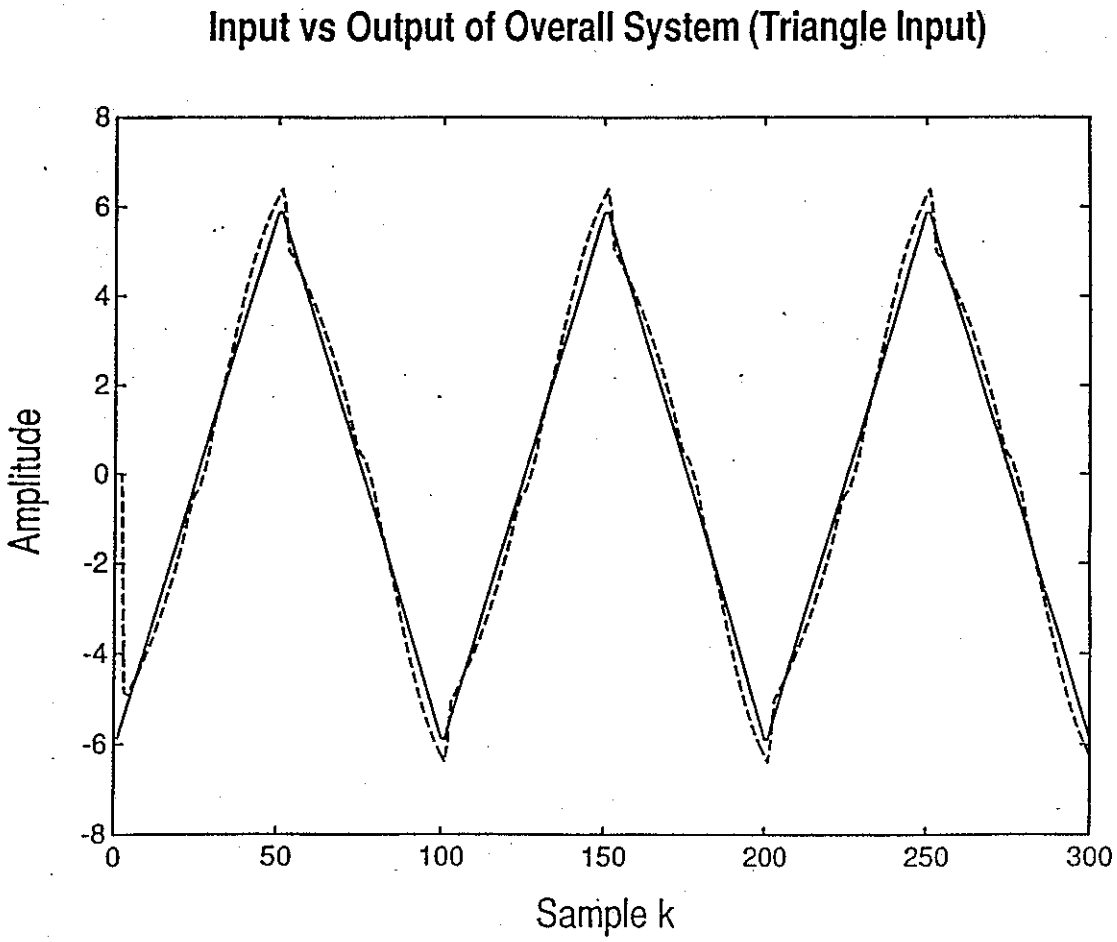

input
1
z -1
1
z -1
1
z -1
a
b
c
d
e
f
g
h
i
1
1
1
output
1
ε
−
error
+
desired
response

Equation of nonlinear plant suggested by K.S. Narendra
(Narendra and Parthasarathy, *IEEE Transactions on Neural Networks*, March 1990.)

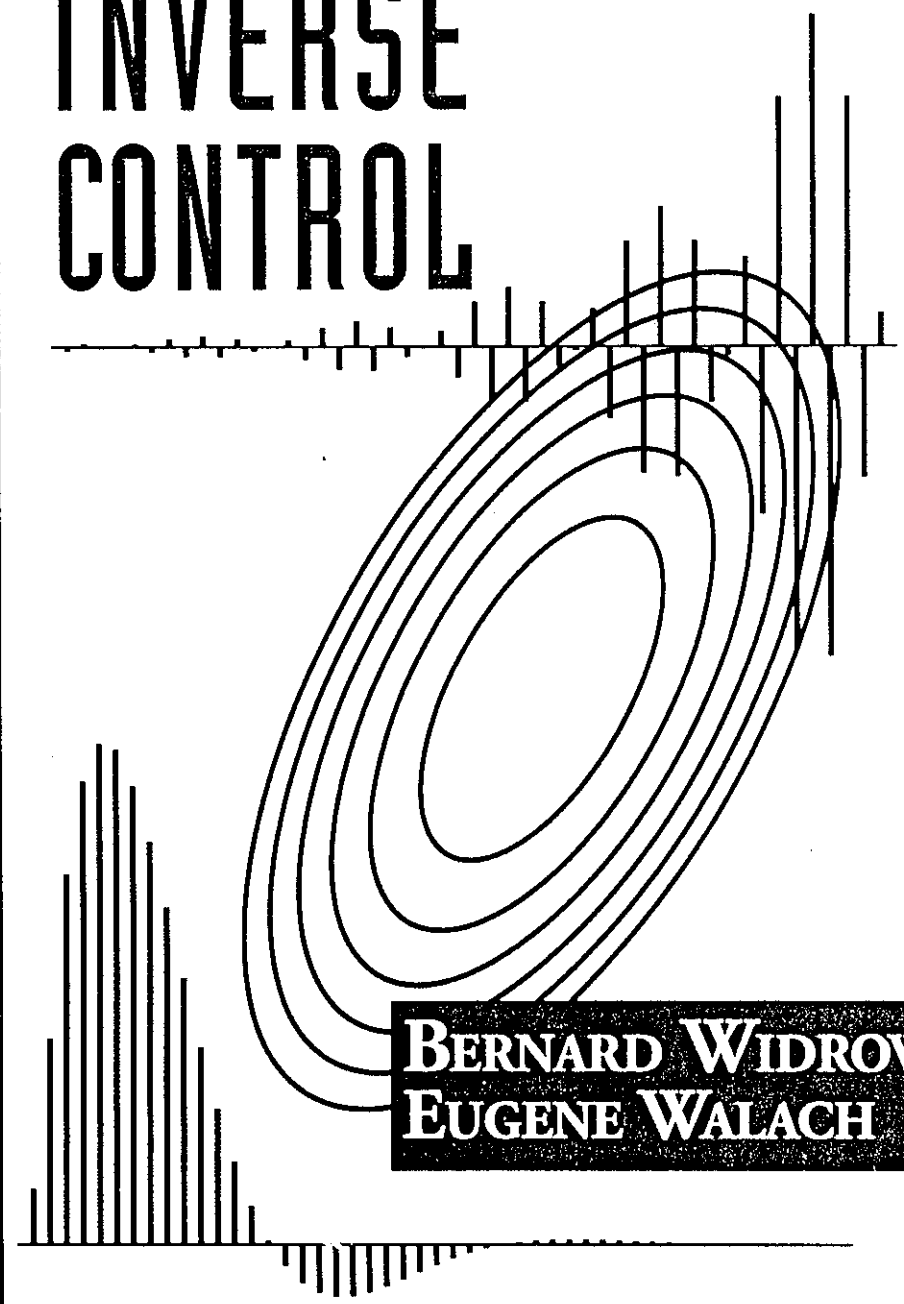
$$y(k) = \frac{y(k-1)}{1 + [y(k-1)]^2} + [u(k-1)]^3$$


Input vs Output of Overall System (Markov 1 Input)
Amplitude
Sample k
5
4
3
2
1
0
-1
-2
-3
0
10
20
30
40
50
60
70
80
90
100


Input vs Output of Overall System (Sine Input)
Amplitude
Sample k
8
6
4
2
0
-2
-4
-6
-8
0
50
100
150
200
250
300


Input vs Output of Overall System (Triangle Input)
Amplitude
Sample k
8
6
4
2
0
-2
-4
-6
-8
0
50
100
150
200
250
300

ADAPTIVE INVERSE CONTROL



BERNARD WIDROW
EUGENE WALACH

PRENTICE HALL INFORMATION & SYSTEM SCIENCES SERIES

Neurointerfaces

Bernard Widrow, *Life Fellow, IEEE,* and Marcelo Malini Lamego, *Member, IEEE*

***Abstract*—A neurointerface is a nonlinear filtering system based on neural networks (NNs) that serves as a coupler between a human operator and a nonlinear system or plant that is to be controlled or directed. The purpose of the coupler is to ease the task of the human controller. The equations of the plant are assumed to be known. If the plant is unstable, it must first be stabilized by feedback. Using the plant equations, off-line automatic learning algorithms are developed for training the weights of the neurointerface and the weights of an adaptive plant disturbance canceller. Application of these ideas to backing a truck with two trailers under human direction is described. The "truck backer" has been successfully demonstrated by computer simulation and by physical implementation with a small radio-controlled truck and trailers.**

***Index Terms*—Adaptive control, man–machine interfaces, neural networks (NNs).**

I. Introduction

FOR MANY tasks, productivity, safety, and liability conditions require a considerable degree of skill from human operators. In order to overcome lack of skill, special man–machine interfaces may be adopted. The basic idea is to change the operational space through a neural network (NN) [1]–[4], allowing the human operator to interact with the process through less-specialized commands. Hence, the operator devotes his attention to solving a less complex problem, directly at the task level. The objective is to improve the productivity and safety levels of such tasks even in the case of unskilled operators.

This paper intends to show how NNs can be applied to the design of man–machine interfaces for practical real-time problems. The term "neurointerface" is chosen to emphasize the use of NNs for the solution of man–machine interface problems. Neurointerfaces can be regarded as circuitries, algorithms, and devices implementing NNs to facilitate the human operation of complex systems.

The design of neurointerfaces involves the training of NNs, which are incorporated in nonlinear adaptive filters. This work applies the inverse modeling technique to designing neurointerfaces. In the past, many works have described training procedures and design techniques for inverse modeling using NNs (see [5], [6], and references therein). This study will apply the inverse modeling technique to the design of man–machine interfaces for complex dynamic systems.

Manuscript received August 29, 2000; revised July 14, 2001. Manuscript received in final form August 24, 2001. Recommended by Associate Editor D. W. Repperger.

B. Widrow is with the Electrical Engineering Department, Stanford University, Stanford CA 94305 USA (e-mail: widrow@stanford.edu).

M. M. Lamego is with The Boston Consulting Group Brazil, Ltd., CEP 01451–0000 Sao Paulo-SP, Brazil (e-mail: mmlamego@ieee.org).

Publisher Item Identifier S 1063-6536(02)00335-4.

In the work of Narendra and Parthasarathy [7], NNs were proposed for the identification and control of nonlinear dynamic systems. As a result, a considerable number of papers have appeared on the general subject of control with NNs. Training schemes based on the backpropagation algorithm [8] and its variations were proposed and applied to NN control.

This work applies dynamic optimization [9], [10] to the training of a multilayer NN with a tapped delay line (neurointerface) cascaded with the plant model. The neurointerface training is formulated as a constrained optimization problem and is solved through dynamic optimization. The technique assumes the nonlinear plant is continuous, minimum phase, and controllable with bounded states (Lagrangian stability). If the plant is unstable, it must first be stabilized by feedback. The backpropagation algorithm is used in one of the steps of the neurointerface training method. For the interested reader, the application of dynamic optimization to state-feedback control with NNs can be found in the works of Lamego [11], Shen [12], and Plumer [13].

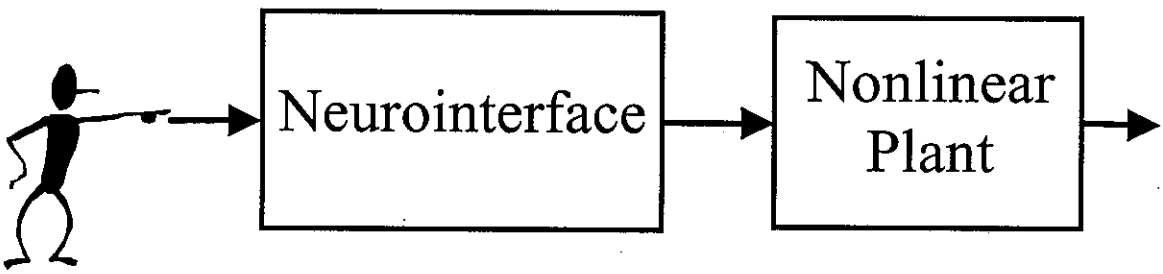
The neurointerface training is done off-line. This is mainly because the dynamic optimization algorithm is computationally very expensive and it also needs the plant model. In addition, depending on the system (plant model plus neurointerface) under study, the time of convergence may be too lengthy for real-time adaptive schemes. Nonetheless, the neurointerface design can be often extended to integrated off-line schemes, wherein a system identification algorithm obtains a continuous representation of the plant model (probably, using an NN) from on-line acquired data. Then, the dynamic optimization algorithm uses the plant model for training the neurointerface, which in turn controls the plant. The processes of plant identification and neurointerface training can be repeated periodically using real-time acquired data.

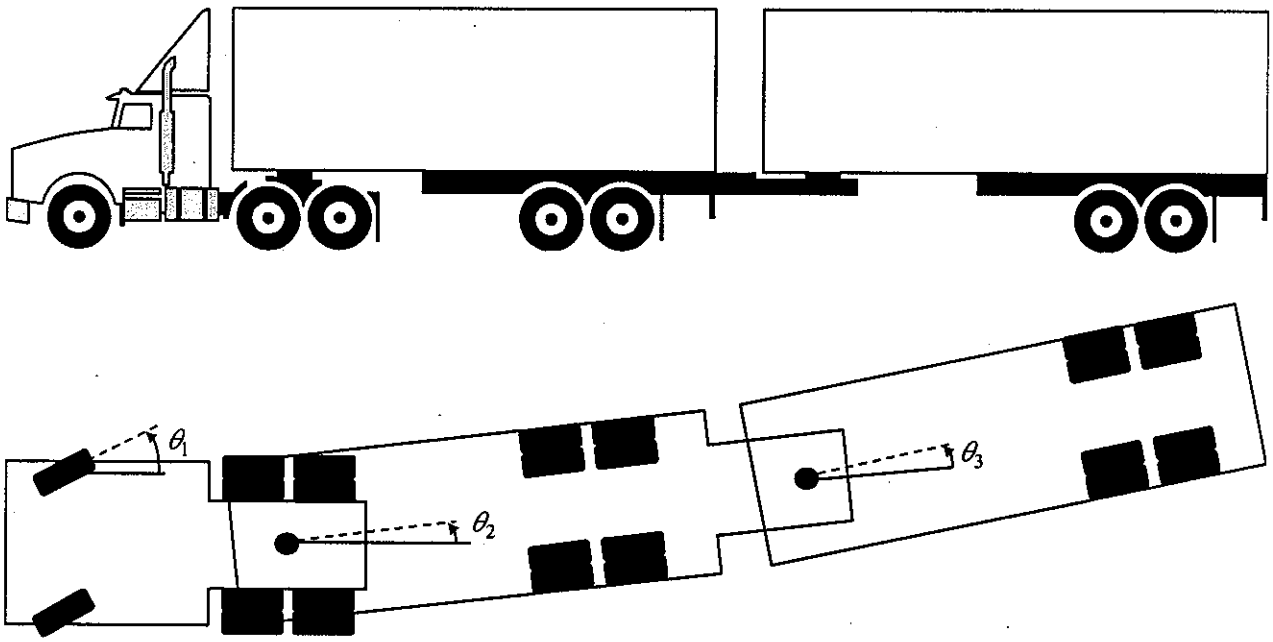
In order to keep the neurointerface training description simple, this paper focus solely on pure off-line training. Once the basic concepts related to plant inversion and dynamic optimization applied to NNs are understood, extensions to real-time learning schemes can be readily made.

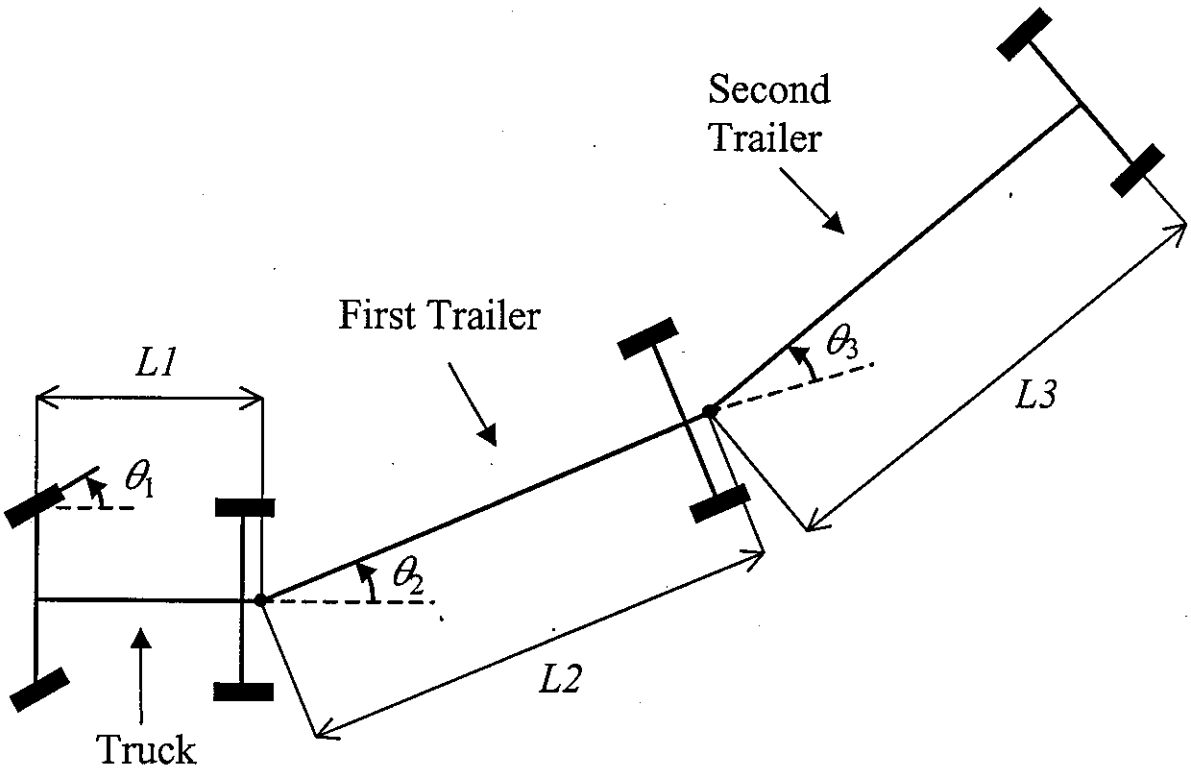
A neurointerface is used to facilitate the backing of a truck connected to a double-trailer configuration under human steering control. The steering commands of the human driver are fed to the neurointerface whose output controls the steering angle of the front wheels of the truck.

II. What is a Neurointerface?

A neurointerface may be thought of as a form of inverse of the plant to be controlled. The desired plant response can be realized by driving the plant with an inverse controller whose input consists of simple command signals applied by a human

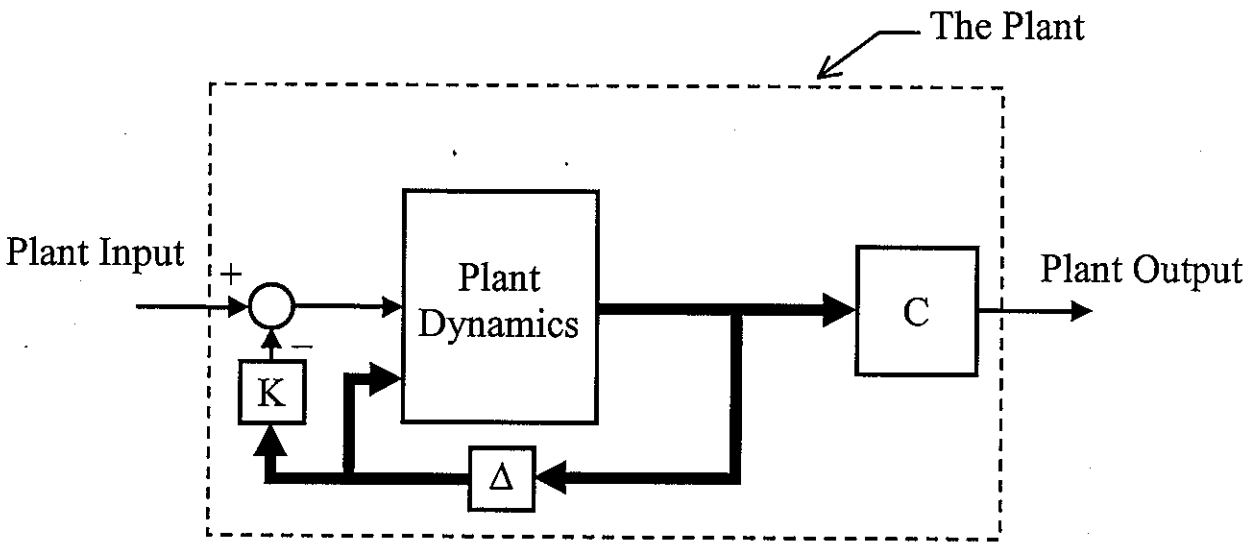

Neurointerface
Nonlinear
Plant

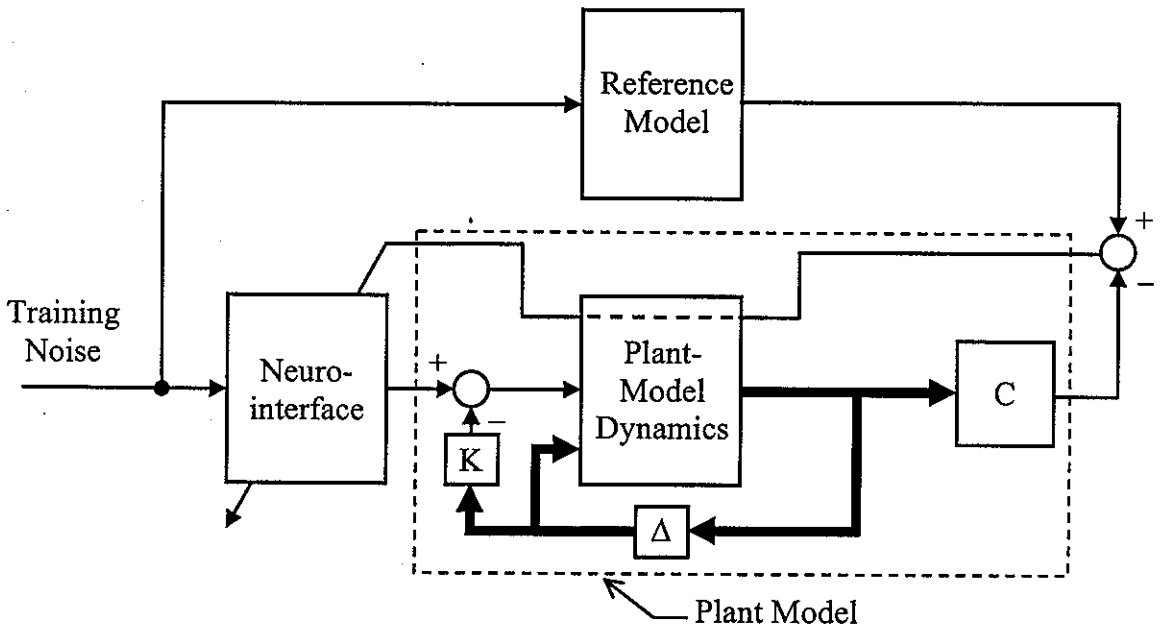

θ_1
θ_2
θ_3

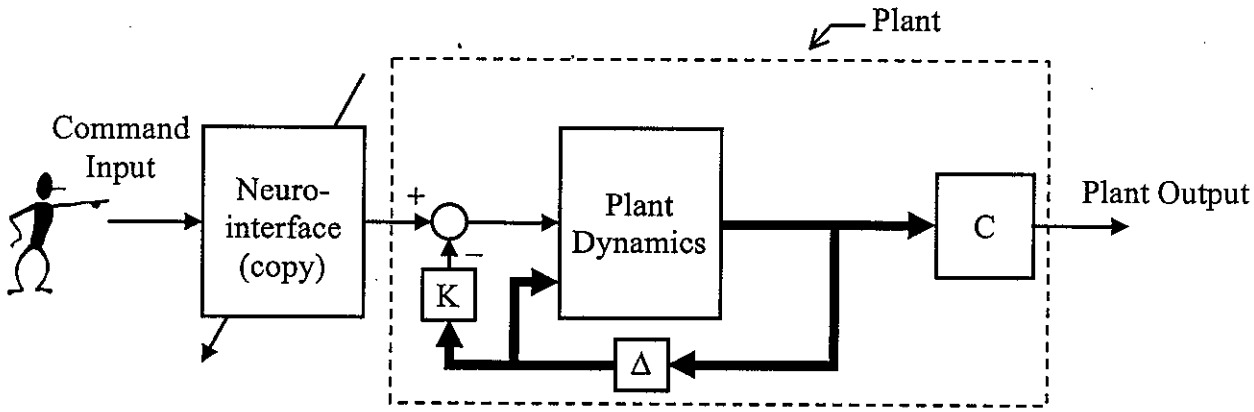


$$\frac{\partial \theta_2}{\partial t} = v\left(\frac{1}{L2}\sin\theta_2 + \frac{1}{L1}\tan\theta_1\right)$$

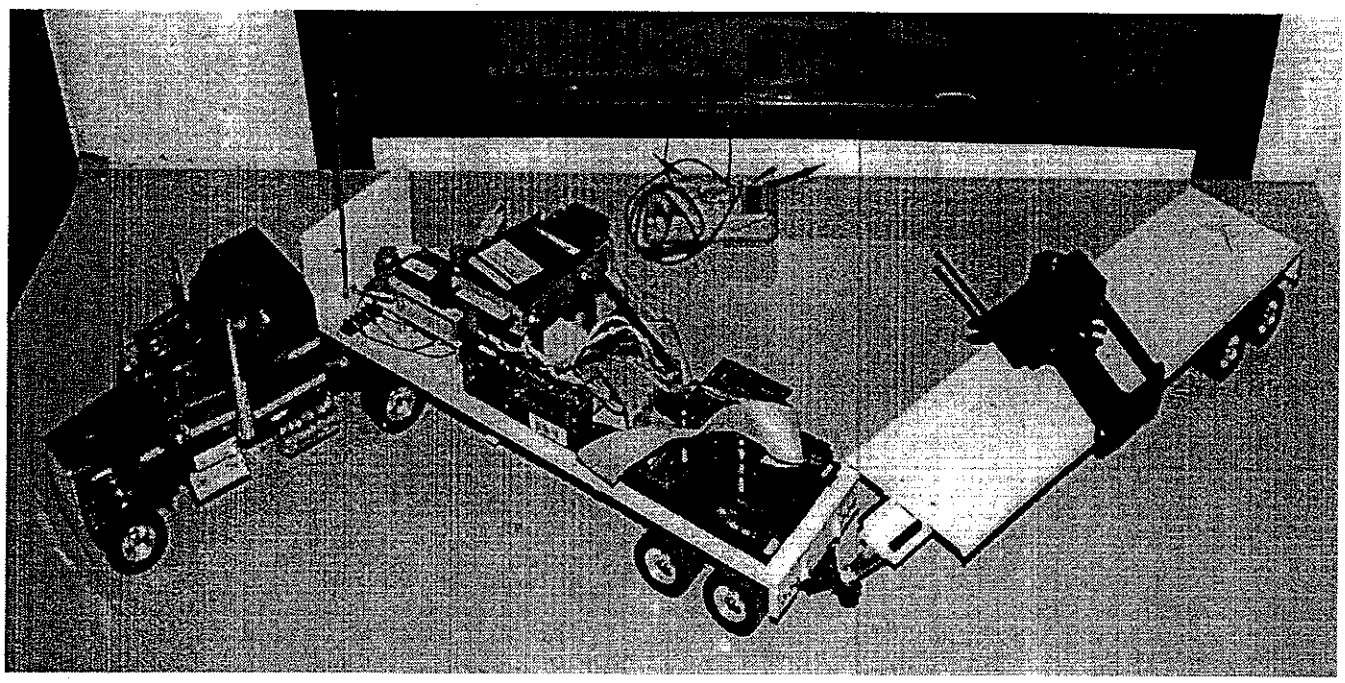
$$\frac{\partial \theta_3}{\partial t} = v\left(-\frac{1}{L2}\sin\theta_2 + \frac{1}{L3}\sin\theta_3\cos\theta_2\right)$$


The Plant
Plant Input
+
−
K
Plant
Dynamics
C
Plant Output
Δ

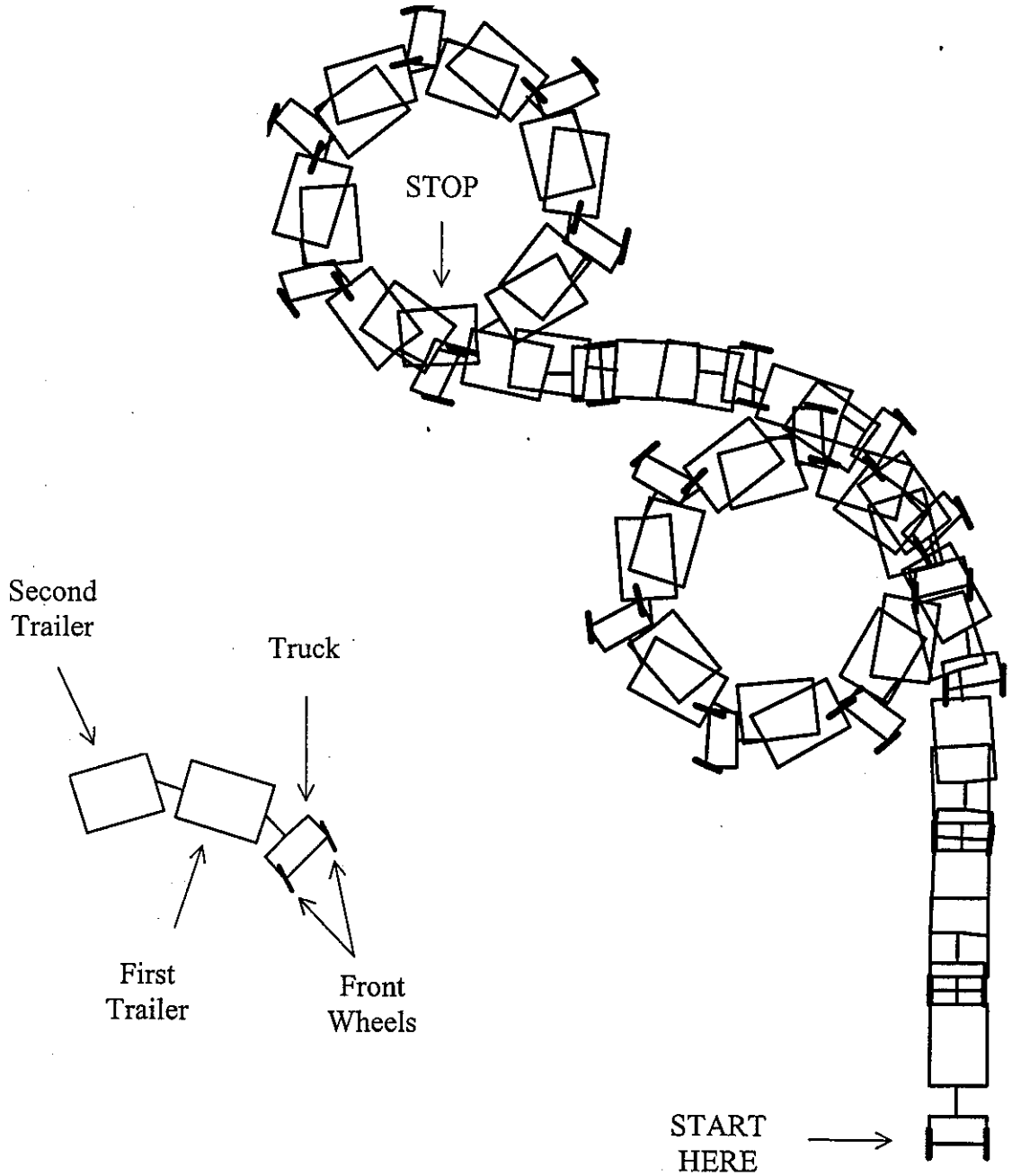

Reference Model
Training Noise
Neuro-interface
+
−
K
Plant-Model Dynamics
C
+
−
Δ
Plant Model
OFF-LINE PROCESS


Plant
Command
Input
Neuro-
interface
(copy)
+
−
K
Plant
Dynamics
C
Plant Output
Δ

Experimental Results



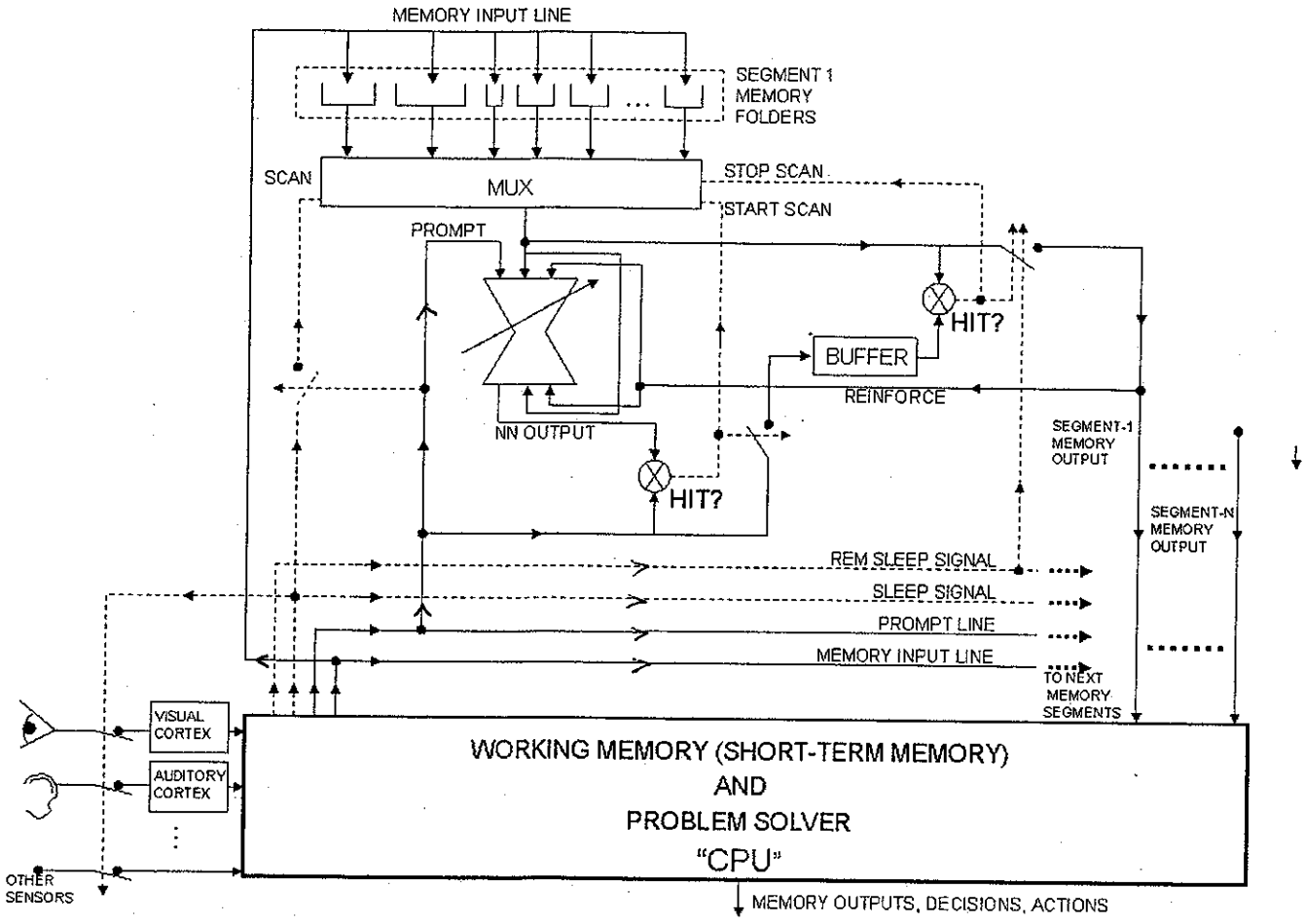
Truck model connected to double-trailer configuration.


STOP
Second Trailer
Truck
First Trailer
Front Wheels
START HERE

"COGNITIVE" MEMORY

DR. BERNARD WIDROW
JUAN CARLOS ARAGON

A "COGNITIVE" MEMORY SYSTEM



SALIENT FEATURES OF THE COGNITIVE MEMORY

•STORES SENSORY PATTERNS (VISUAL, AUDITORY, TACTILE; RADAR, SONAR, ETC.)

•STORES PATTERNS WHEREVER SPACE IS AVAILABLE, NOT IN SPECIFIED MEMORY LOCATIONS

•STORES SIMULTANEOUSLY SENSED INPUT PATTERNS IN THE SAME FOLDER
(e.g. SIMULTANEOUS VISUAL AND AUDITORY PATTERNS ARE STORED TOGETHER)

•DATA RECOVERY IS IN RESPONSE TO "PROMPT" INPUT PATTERNS
(e.g. A VISUAL OR AUDITORY INPUT PATTERN WOULD TRIGGER RECALL)

•AUTOASSOCIATIVE NEURAL NETWORKS ARE USED IN THE DATA RETRIEVAL SYSTEM

GOALS

- DEVELOP AND REFINE COGNITIVE MEMORY CONCEPT
  - RELATE TO HUMAN MEMORY
  - DETERMINE MEMORY CAPACITY
  - OPTIMIZE DESIGN OF AUTOASSOCIATIVE NEURAL NETWORKS

- DEVELOP APPLICATIONS FOR COGNITIVE MEMORY

- DEVELOP HARDWARE PLATFORM FOR HIGH-SPEED AND HIGH-CAPACITY COGNITIVE MEMORY

STATUS

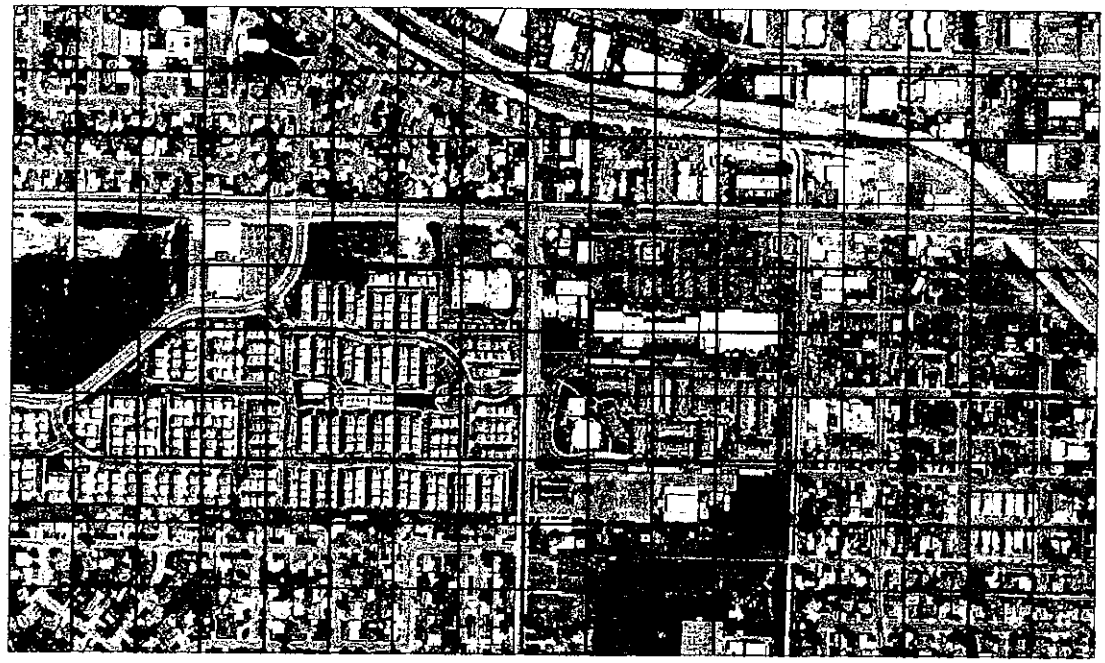
•WE HAVE SIMULATED A WORKING COGNITIVE MEMORY THAT EMULATES CERTAIN CHARACTERISTICS OF HUMAN MEMORY

•WE HAVE MADE PRELIMINARY STUDIES OF CAPACITY AND SPEED

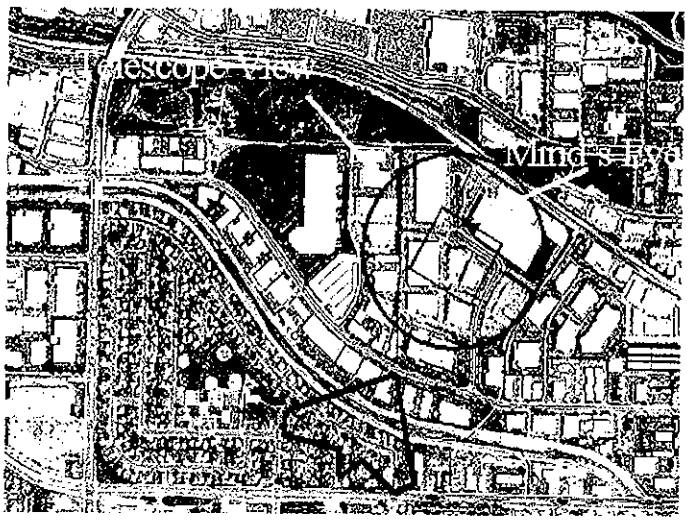
•WE HAVE DEVELOPED FOUR PRELIMINARY APPLICATIONS FOR THE COGNITIVE MEMORY

•WE ARE DEVELOPING A DESIGN FOR THE AUTOASSOCIATIVE NEURAL NETWORK BASED ON FPGA TECHNOLOGY THAT WOULD ALLOW VERY HIGH-SPEED OPERATION

•WE ARE CONTINUING RESEARCH IN ALL THESE AREAS



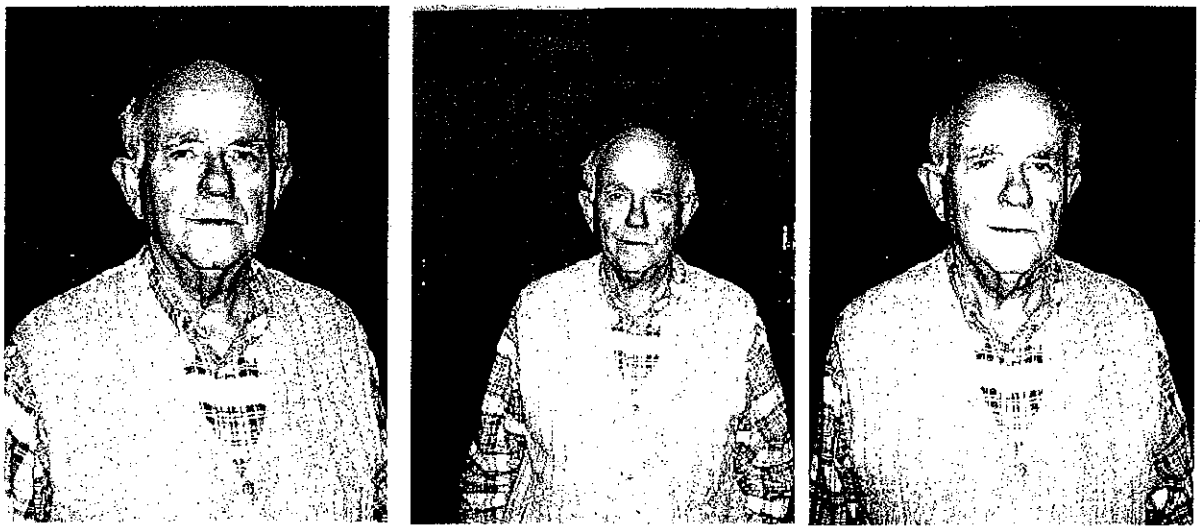
Diced aerial photo of Simi Valley, CA



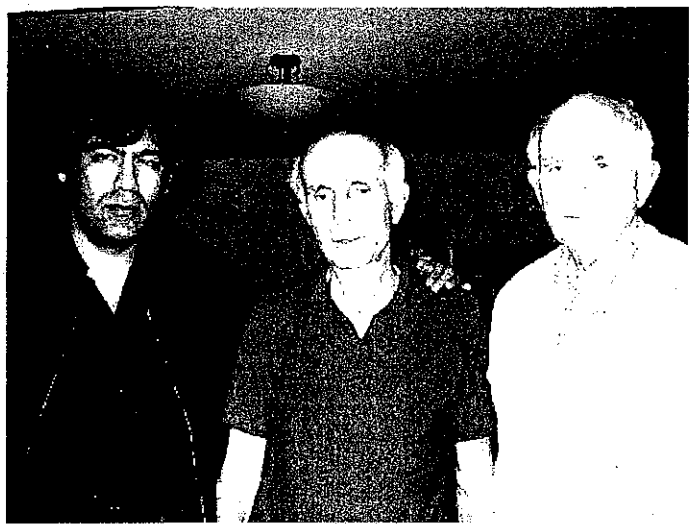
Shooting the ground
through a telescope to find
airplane's position

Circular airplane track over Simi Valley, CA

THREE PHOTOS OF BERNARD WIDROW USED FOR TRAINING



A PHOTO OF JUAN CARLOS ARAGON, VICTOR ELIASHBERG, AND BERNARD WIDROW USED FOR SENSING



FACIAL DETECTION

TRAINING (LOW RESOLUTION, 20x20 pixel images)

- ONE IMAGE OF A PERSON'S FACE WAS TRAINED IN
- THAT IMAGE WAS ADJUSTED BY
  - ROTATION (2 DEGREE INCREMENTS, 7 ANGLES)
  - TRANSLATION (LEFT/RIGHT, UP/DOWN, 1 PIXEL INCREMENTS, 9 POSITIONS)
  - BRIGHTNESS (5 LEVELS OF INTENSITY)

- TOTAL NUMBER OF TRAINING PATTERNS = 315

- TRAINING TIME 2 HOURS ON AMD 64 BIT ATHLON 2.6 GHZ COMPUTER FOR 0.25 PERCENT MSE

FACIAL DETECTION

SENSING (LOW RESOLUTION, 20x20 pixel images)

- EACH INPUT PATTERN WAS ADJUSTED BY
  - SCALING (6 WINDOW SIZES)
  - TRANSLATION (90 PIXEL INCREMENTS)

- ERRORS WITH BACKGROUND WERE ABOUT 8X GREATER THAN WITH A PERSON'S FACE

- 60 PATTERNS PER SECOND THROUGH NEURAL NETWORK

- AUTOASSOCIATIVE NEURAL NETWORK HAD A TOTAL OF 1100 NEURONS DISTRIBUTED OVER 3 LAYERS
  - 400 NEURONS, 400 WEIGHTS PER NEURON, FIRST LAYER
  - 300 NEURONS, 400 WEIGHTS PER NEURON, SECOND LAYER
  - 400 NEURONS, 300 WEIGHTS PER NEURON, THIRD LAYER

FACIAL RECOGNITION

TRAINING (HIGH RESOLUTION, 50x50 pixel images)

- ALL THREE IMAGES OF WIDROW'S FACE WERE TRAINED IN
- EACH IMAGE WAS ADJUSTED BY
  - ROTATION (2 DEGREE INCREMENTS, 7 ANGLES)
  - TRANSLATION (LEFT/RIGHT, UP/DOWN, 1 PIXEL INCREMENTS, 25 POSITIONS)
  - SCALING (3 WINDOW SIZES)

- TOTAL NUMBER OF TRAINING PATTERNS = 1575

- TRAINING TIME 26 HOURS ON AMD 64 BIT ATHLON 2.6 GHZ COMPUTER FOR 0.25 PERCENT MSE

FACIAL RECOGNITION

SENSING (HIGH RESOLUTION, 50x50 pixel images)

- EACH INPUT PATTERN WAS ADJUSTED BY
  - SCALING (6 WINDOW SIZES)
  - TRANSLATION (2 PIXEL INCREMENTS, 25 POSITIONS)
  - BRIGHTNESS (6 LEVELS OF INTENSITY)

- OPTIMIZATION WAS DONE FOR EACH DETECTED FACE

- ERRORS WITH UNIDENTIFIED FACES WERE ABOUT 4X GREATER THAN WITH DR. WIDROW'S FACE

- 5 PATTERNS PER SECOND THROUGH NEURAL NETWORK

- AUTOASSOCIATIVE NEURAL NETWORK
  - 1800 NEURONS, 2500 WEIGHTS PER NEURON, FIRST LAYER
  - 1500 NEURONS, 1800 WEIGHTS PER NEURON, SECOND LAYER
  - 2500 NEURONS, 1500 WEIGHTS PER NEURON, THIRD LAYER
  - TOTAL 5800 NEURONS, 10,950,000 WEIGHTS

COGNITIVE MEMORY DEMONSTRATION
FACE RECOGNITION






Balvani


© Dhar Balvani


© Dhar Balvani


© Dhar Balvani