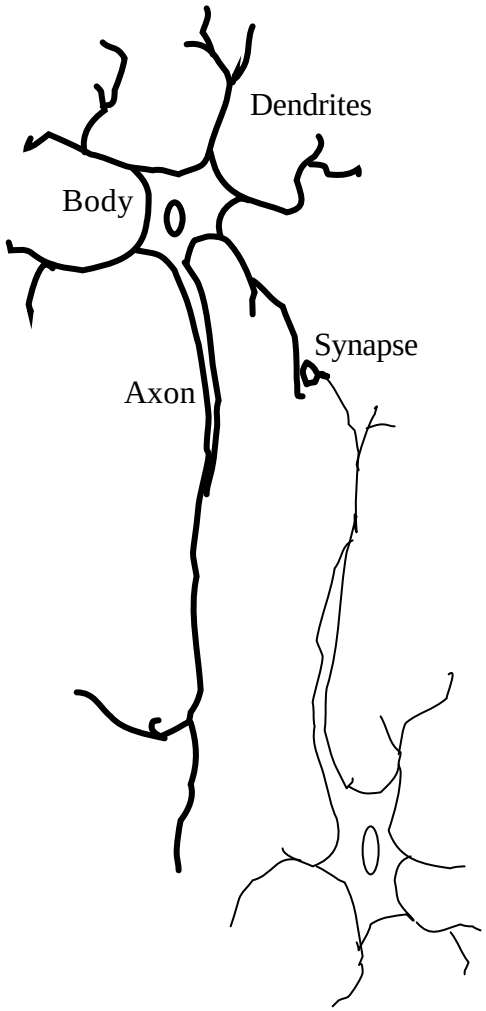
NEURAL NETWORKS FOR MODELLING APPLICATIONS

Emil M. Petriu, Dr. Eng., P. Eng., FIEEE
Professor
School of Information Technology and Engineering
University of Ottawa
Ottawa, ON., Canada
http://www.site.uottawa.ca/~petriu/
petriu@site.uottawa.ca

Biological Neurons



- **Dendrites** carry electrical signals in into the neuron body.
  The neuron **body** integrates and thresholds the incoming signals.
  The **axon** is a single long nerve fiber that carries the signal from the neuron body to other neurons.
  A **synapse** is the connection between dendrites of two neurons.

- Incoming signals to a dendrite may be inhibitory or excitatory. The strength of any input signal is determined by the strength of its synaptic connection. A neuron sends an impulse down its axon if excitation exceeds inhibition by a critical amount (threshold/ offset/bias) within a time window (period of latent summation).

- *Memories* are formed by the modification of the **synaptic strengths** which can change during the entire life of the neural systems..

- Biological neurons are rather slow (10^{-3} s) when compared with the modern electronic circuits. ==> The brain is faster than an electronic computer because of its massively parallel structure. The brain has approximately 10^{11} highly connected neurons (approx. 10^4 connections per neuron).

Historical Sketch of Neural Networks

1940s

Natural components of mind-like machines are simple abstractions based on the behavior of biological nerve cells, and such machines can be built by interconnecting such elements.

- **W. McCulloch** & **W. Pitts** (1943) the first theory on the fundamentals of neural computing (neuro-logicalnetworks) "A Logical Calculus of the Ideas Immanent in Nervous Activity" ==> *McCulloch-Pitts neuron model*; (1947) "How We Know Universals" - an essay on networks capable of recognizing spatial patterns invariant of geometric transformations.

- **Cybernetics:** attempt to combine concepts from biology, psychology, mathematics, and engineering.

- **D.O. Hebb** (1949) "The Organization of Behavior" the first theory of psychology on conjectures about neural networks (neural networks might learn by constructing internal representations of concepts in the form of "cell-assemblies" - subfamilies of neurons that would learn to support one another's activities). ==> *Hebb's learning rule*: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

1950s

Cybernetic machines developed as specific architectures to perform specific functions.
==> "machines that could learn to do things they aren't built to do"

- **M. Minsky** (1951) built a reinforcement-based network learning system.

- **IRE Symposium "The Design of Machines to Simulate the Behavior of the Human Brain"** (1955) with four panel members: W.S. McCulloch, A.G. Oettinger, O.H. Schmitt, N. Rochester, invited questioners: M. Minsky, M. Rubinoff, E.L. Gruenberg, J. Mauchly, M.E. Moran, W. Pitts, and the moderator H.E. Tompkins.

- **F. Rosenblatt** (1958) the first practical Artificial Neural Network (ANN) - the *perceptron*, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.".

⇨ By the end of 50s, the NN field became dormant because of the new AI advances based on serial processing of symbolic expressions.

1960s

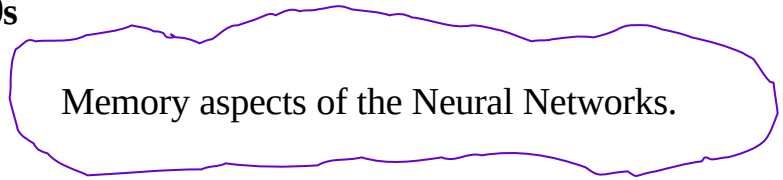
Connectionism (Neural Networks) - versus - **Symbolism** (Formal Reasoning)

- **B. Widrow** & **M.E. Hoff** (1960) "Adaptive Switching Circuits" presents an adaptive percepton-like network. The weights are adjusted so to minimize the mean square error between the actual and desired output ==> *Least Mean Square (LMS) error algorithm.* (1961) Widrow and his students "Generalization and Information Storage in Newtworks of Adaline "Neurons."

- **M. Minsky** & **S. Papert** (1969) "Perceptrons" a formal analysis of the percepton networks explaining their limitations and indicating directions for overcoming them ==> *relationship between the perceptron's architecture and what it can learn*: "no machine can learn to recognize X unless it poses some scheme for representing X."

  Limitations of the perceptron networks led to the pessimist view of the NN field as having no future ==> no more interest and funds for NN research!!!

1970s

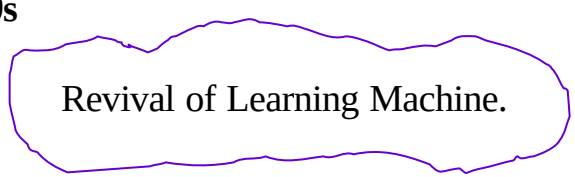


- **T. Kohonen** (1972) "Correlation Matrix Memories" a mathematical oriented paper proposing a correlation matrix model for associative memory which is trained, using Hebb's rule, to learn associations between input and output vectors.

- **J.A. Anderson** (1972) "A Simple Neural Network Generating an Interactive Memory" a physiological oriented paper proposing a "linear associator" model for associative memory, using Hebb's rule, to learn associations between input and output vectors.

- **S. Grossberg** (1976) "Adaptive Pattern Classification and Universal Recording: I. Parallel Development and Coding of Neural Feature Detectors"describes a self-organizing NN model of the visual system consisting of a short-term and long term memory mechanisms. ==> continuous-time competitive network that forms a *basis for the Adaptive Resonance Theory (ART) networks*.

1980s



> [Minsky]: "*The marvelous powers of the brain emerge not from any single, uniformly structured connectionst network but from highly evolved arrangements of smaller, specialized networks which are interconnected in very specific ways.*"

- **D.E. Rumelhart** & **J.L. McClelland**, eds. (1986) "Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Explorations in the Microstructure of Cognition" represents a milestone in the resurgence of NN research.

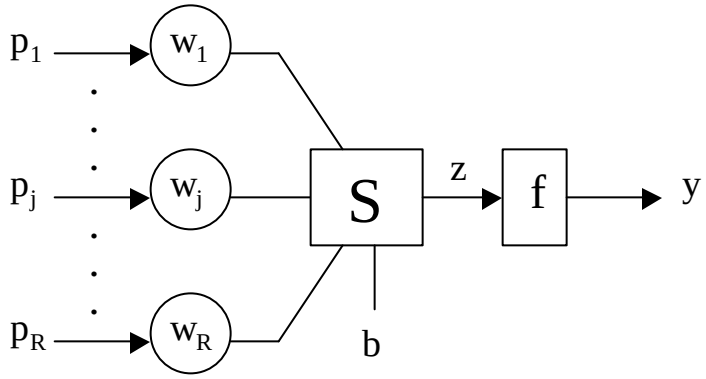
- **J.A. Anderson** & **E. Rosenfeld** (1988) "Neurocomputing: Foundations of Research" contains over forty seminal papers in the NN field.

- **DARPA Neural Network Study**(1988) a comprehensive review of the theory and applications of the Neural Networks.

- **International Neural Network Society** (1988) …. **IEEE Tr. Neural Networks** (1990).

Artificial Neural Networks (ANN)

McCulloch-Pitts model of an artificial neuron



$$y = f\left(w_1 \cdot p_1 + \ldots + w_j \cdot p_j + \ldots w_R \cdot p_R + b \right)$$

$$y = f\left(\boldsymbol{W} \cdot \boldsymbol{p} + b \right)$$

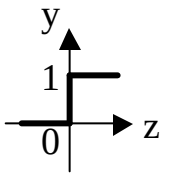
$\boldsymbol{p} = (p_1, \ldots, p_R)^T$ is the input column-vector

$\boldsymbol{W} = (w_1, \ldots, w_R)$ is the weight row-vector

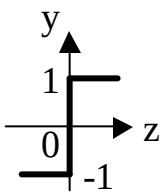
*) The bias *b* can be treated as a weight whose input is always 1.

Some transfer functions "**f**"

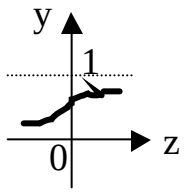
Hard Limit: y = 0 if z<0
y = 1 if z>=0



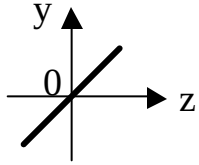
Symmetrical Hard Limit: y = -1 if z<0
y = +1 if z>=0



Log-Sigmoid:
$y = 1/(1+e^{-z})$



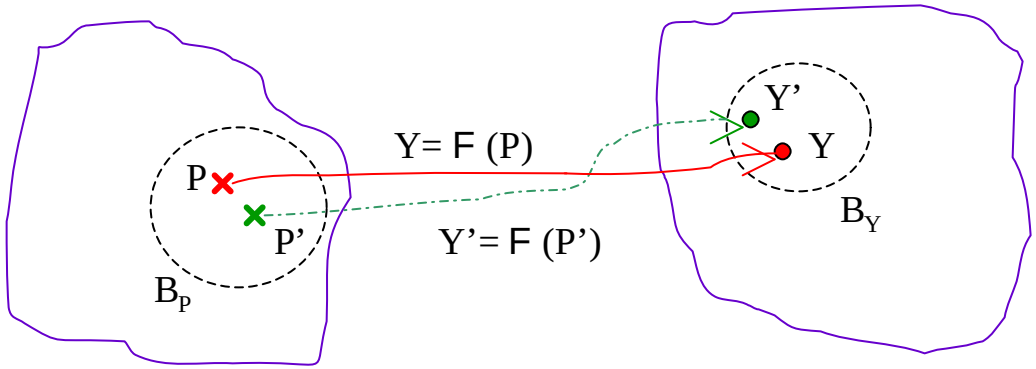
Linear:
y = z



The Architecture of an ANN

- Number of inputs and outputs of the network;
- Number of layers;
- How the layers are connected to each other;
- The transfer function of each layer;
- Number of neurons in each layer;

ANNs map input/stimulus values to output/response values: $Y = F(P)$.

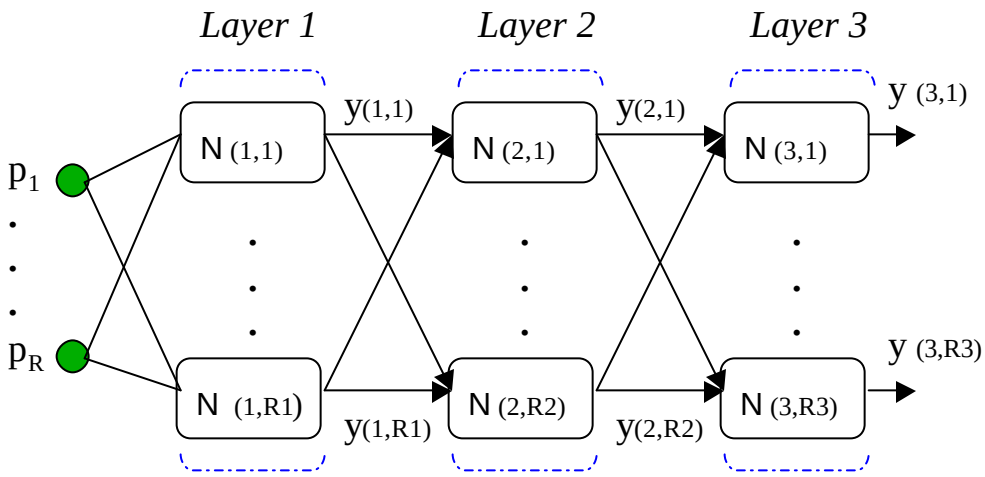


Intelligent systems generalize: their behavioral repertoires exceed their experience. An intelligent system is said to have a creative behaviour if it provides appropriate responses when faced with new stimuli. Usually the new stimuli *P'* resemble known stimuli *P* and their corresponding responses *Y'* resemble known/learned responses *Y*.

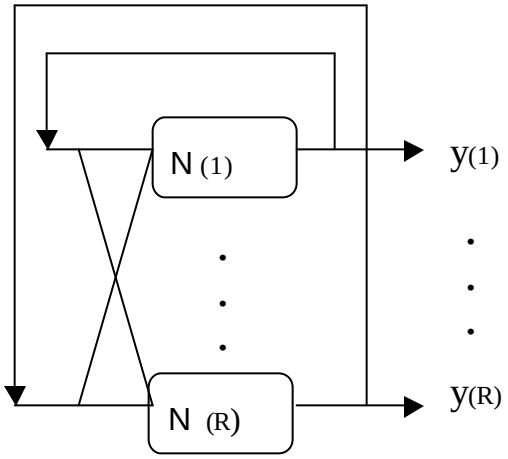
Measure of system's F creativity:

$$\frac{\text{Volume of "stimuli ball } B_P \text{ "}}{\text{Volume of "response ball } B_Y \text{"}}$$

- Most of the mapping functions can be implemented by a two-layer ANN: a sigmoid layer feeding a linear output layer.
- ANNs with biases can represent relationships between inputs and outputs than networks without biases.
- *Feed-forward* ANNs cannot implement temporal relationships. *Recurrent* ANNs have internal feedback paths that allow them to exhibit temporal behaviour.



Feed-forward architecture with three layers



Recurrent architecture (*Hopfield* NN)

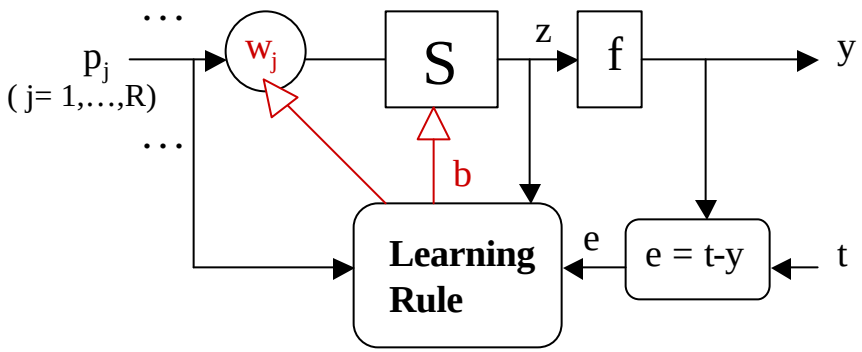
The ANN is usually supplied with an initial input vector and then the outputs are used as inputs for each succeeding cycle.

Learning Rules (Training Algorithms)

Procedure/algorithm to adjust the weights and biases in order for the ANN to perform the desired task.

Supervised Learning

For a given training set of pairs {***p***(1),t(1)},...,{***p***(n),t(n)}, where ***p***(i) is an instance of the input vector and **t**(i) is the corresponding *target* value for the output ***y***, the learning rule calculates the updated value of the neuron weights and bias.



Reinforcement Learning

Similar to supervised learning - instead of being provided with the correct output value for each given input, the algorithm is only provided with a given grade/score as a measure of ANN's performance.

Unsupervised Learning

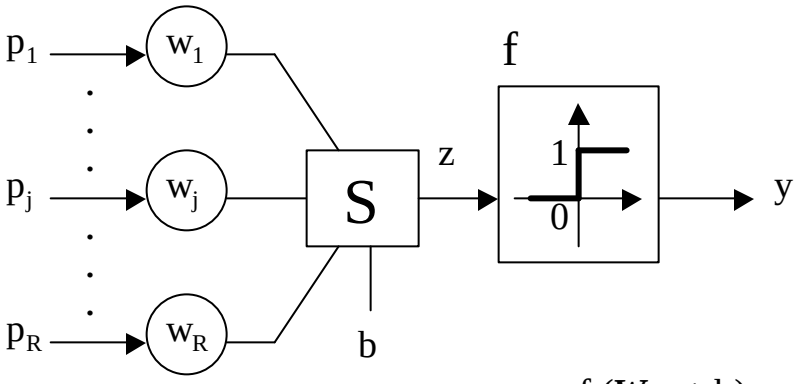
The weight and unbiased are adjusted based on inputs only. Most algorithms of this type learn to cluster input patterns into a finite number of classes. ==> e.g. vector quantization applications

THE PERCEPTRON

- Frank Rosenblatt (1958), Marvin Minski & Seymour Papert (1969)

- [Minski] "Perceptrons make decisions/determine whether or not event fits a certain pattern by adding up evidence obtained from many small experiments"

- The **perceptron** is a neuron with a hard limit transfer function and a weight adjustment mechanism ("learning") by comparing the actual and the expected output responses for any given input /stimulus.



$y = f(\boldsymbol{W} \cdot \boldsymbol{p} + b)$

- Perceptrons are well suited for pattern classification/recognition.

- The weight adjustment/training mechanism is called the *perceptron learning rule*.

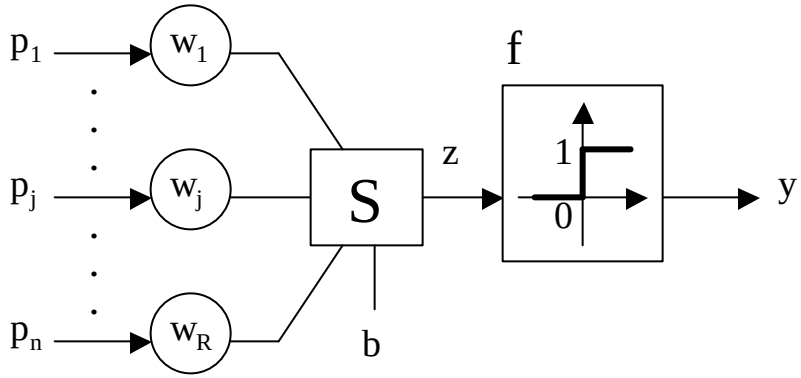
NB: $\boldsymbol{W}$ is a row-vector and $\boldsymbol{p}$ is a column-vector.

Perceptron Learning Rule

- **Supervised learning**

t <== the target value

e = t-y <== the error



Because of the perceptron's hard limit transfer function *y, t, e* can take only binary values

$\boldsymbol{p} = (p_1, \ldots, p_R)^T$ is the input column-vector

$\boldsymbol{W} = (x_1, \ldots, x_R)$ is the weight row-vector

Perceptron learning rule:

if e = 1, then $\boldsymbol{W}^{new} = \boldsymbol{W}^{old} + \boldsymbol{p}$, $b^{new} = b^{old} + 1$;

if e = -1, then $\boldsymbol{W}^{new} = \boldsymbol{W}^{old} - \boldsymbol{p}$, $b^{new} = b^{old} - 1$;

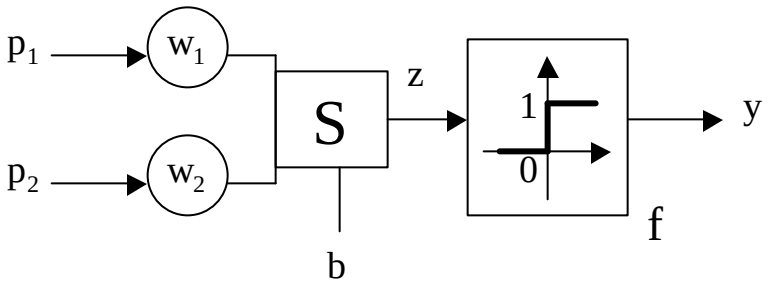
if e = 0, then $\boldsymbol{W}^{new} = \boldsymbol{W}^{old}$.

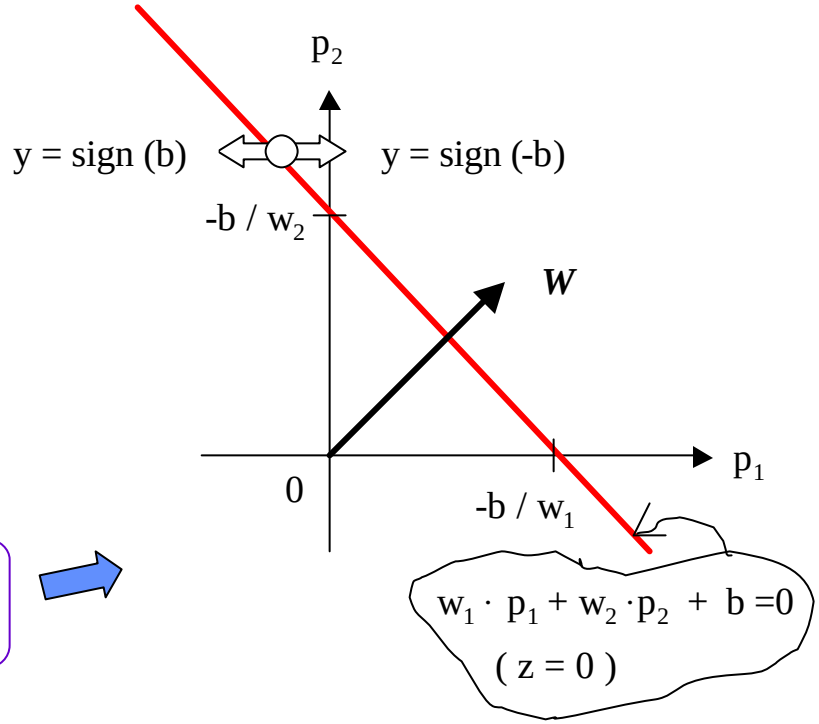
$$\boldsymbol{W}^{new} = \boldsymbol{W}^{old} + e\boldsymbol{p}^T$$

$$b^{new} = b^{old} + e$$

The hard limit transfer function (threshold function) provides the ability to classify input vectors by deciding whether an input vector belongs to one of two *linearly separable classes.*

Two-Input Perceptron





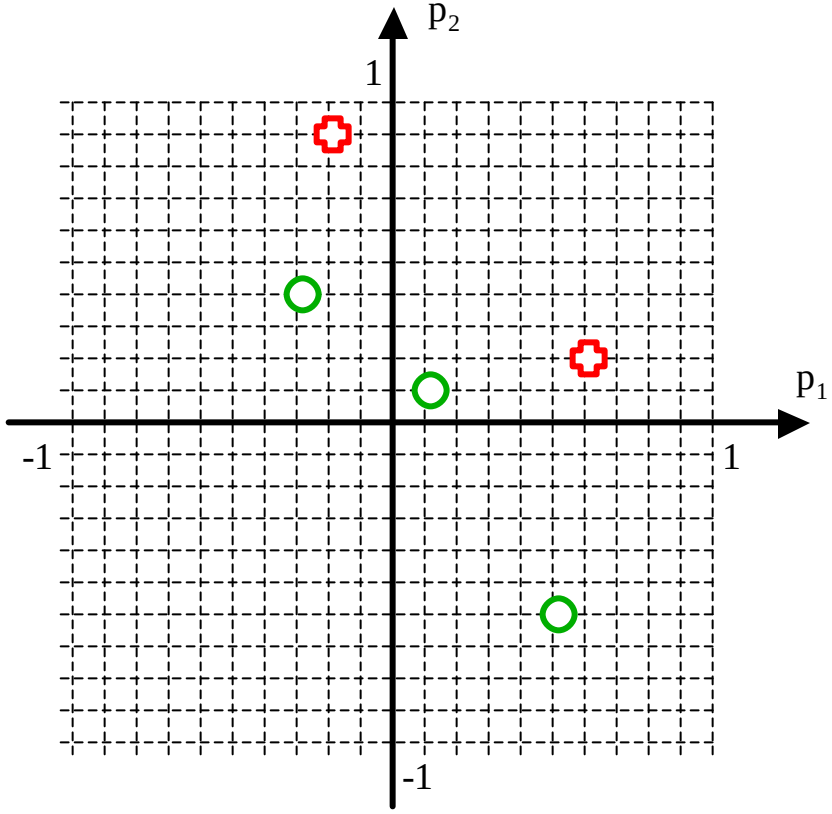
$$y = hardlim\,(z) = hardlim\{\,[w_1\,, w_2] \cdot [p_1\,, p_2]^T + b\}$$

- The two classes (linearly separable regions) in the two-dimensional input space (p_1, p_2) are separated by the line of equation $z = 0$.
- The boundary is always orthogonal to the weight vector $\boldsymbol{W}$.

❑ **Example #1**: Teaching a two-input perceptron to classify five input vectors into two classes

$\left\{ \begin{matrix} \boldsymbol{p}(1) = (0.6, 0.2)^T \\ \boldsymbol{t}(1) = 1 \end{matrix} \right\}$ $\left\{ \begin{matrix} \boldsymbol{p}(2) = (-0.2, 0.9)^T \\ \boldsymbol{t}(2) = 1 \end{matrix} \right\}$ $\left\{ \begin{matrix} \boldsymbol{p}(3) = (-0.3, 0.4)^T \\ \boldsymbol{t}(3) = 0 \end{matrix} \right\}$ $\left\{ \begin{matrix} \boldsymbol{p}(4) = (0.1, 0.1)^T \\ \boldsymbol{t}(4) = 0 \end{matrix} \right\}$ $\left\{ \begin{matrix} \boldsymbol{p}(5) = (0.5, -0.6)^T \\ \boldsymbol{t}(5) = 0 \end{matrix} \right\}$

▶ The MATLAB solution is:




```
P=[0.6 -0.2 -0.3 0.1 0.5;
   0.2  0.9  0.4 0.1 -0.6];
T=[1 1 0 0 0];
W=[-2 2];
b=-1;
plotpv(P,T);
plotpc(W,b);
nepoc=0
Y=hardlim(W*P+b);
while any(Y~=T)
Y=hardlim(W*P+b);
E=T-Y;
[dW,db]= learnp(P,E);
W=W+dW;
b=b+db;
nepoc=nepoc+1;
disp('epochs='),disp(nepoc),
disp(W), disp(b);
plotpv(P,T);
plotpc(W,b);
end
```

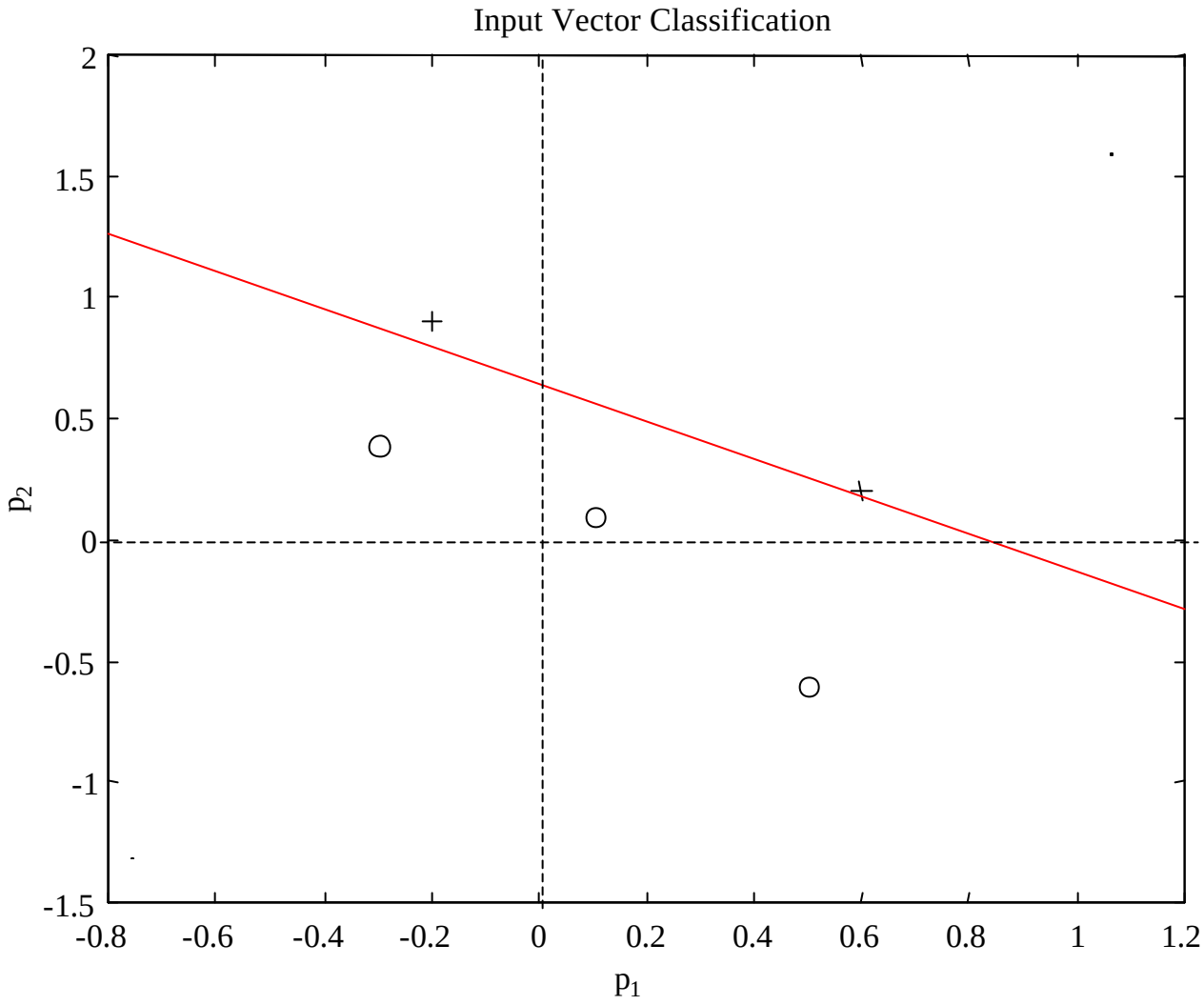
❑ **Example #1**:

After *nepoc* = 11 (epochs of training starting from an initial weight vector `W=[-2 2]` and a `bias b=-1`) the weights are:

$w_1 = 2.4$
$w_2 = 3.1$

and the bias is:

$b = -2$



- The larger an input vector $\boldsymbol{p}$ is, the larger is its effect on the weight vector $\boldsymbol{W}$ during the learning process

  Long training times can be caused by the presence of an "outlier," i.e. an input vector whose magnitude is much larger, or smaller, than other input vectors.

  Normalized perceptron learning rule, the effect of each input vector on the weights is of the same magnitude:

$$\boldsymbol{W}^{new} = \boldsymbol{W}^{old} + e\boldsymbol{p}^T / \|\boldsymbol{p}\|$$

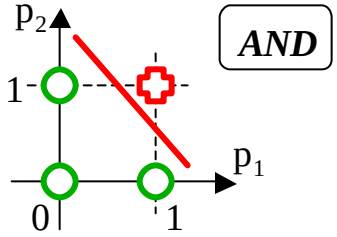
$$b^{new} = b^{old} + e$$

Perceptron Networks for Linearly Separable Vectors

- The hard limit transfer function of the perceptron provides the ability to classify input vectors by deciding whether an input vector belongs to one of two *linearly separable classes.*

$\mathbf{p} = [0\ 0\ 1\ 1;\ 0\ 1\ 0\ 1]$

$\mathbf{t}_{AND} = [0\ 0\ 0\ 1]$

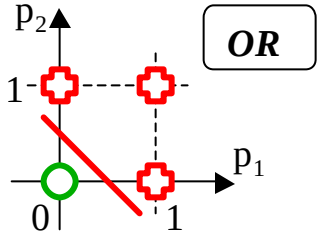


$\mathbf{W} = [2\ \ 2]$

$b = -3$

$\mathbf{p} = [0\ 0\ 1\ 1;\ 0\ 1\ 0\ 1]$

$\mathbf{t}_{OR} = [0\ 1\ 1\ 1]$

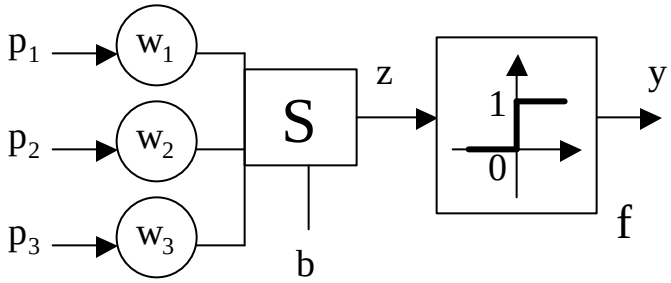


$\mathbf{W} = [2\ \ 2]$

$b = -1$

Three-Input Perceptron

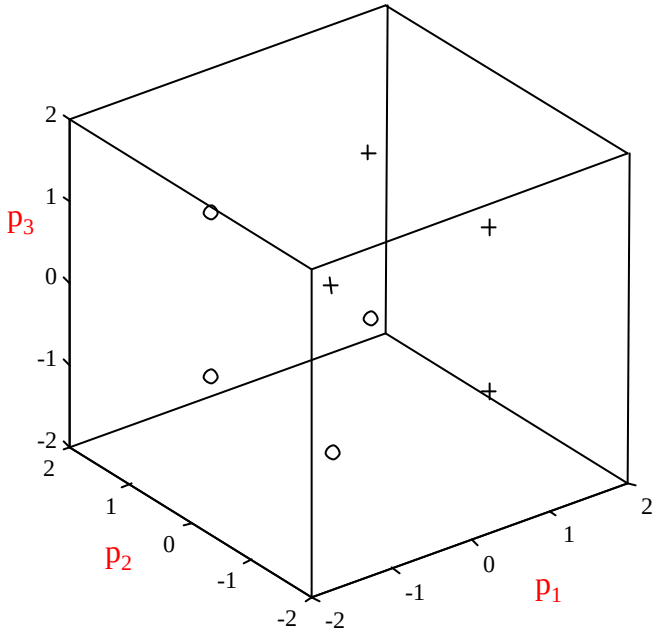
✔ The two classes in the 3-dimensional input space (p_1, p_2, p_3) are separated by the plane of equation $z = 0$.

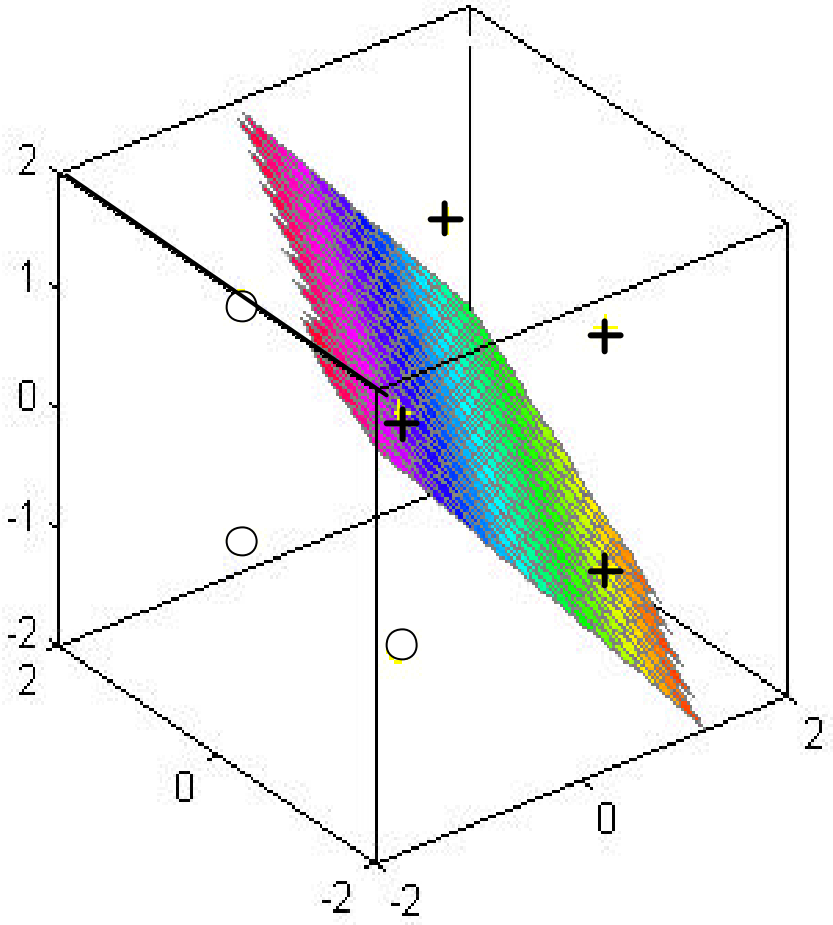


$$y = hardlim(z) = hardlim\{[w_1, w_2, w_3] \cdot [p_1, p_2 p_3]^T + b\}$$

EXAMPLE

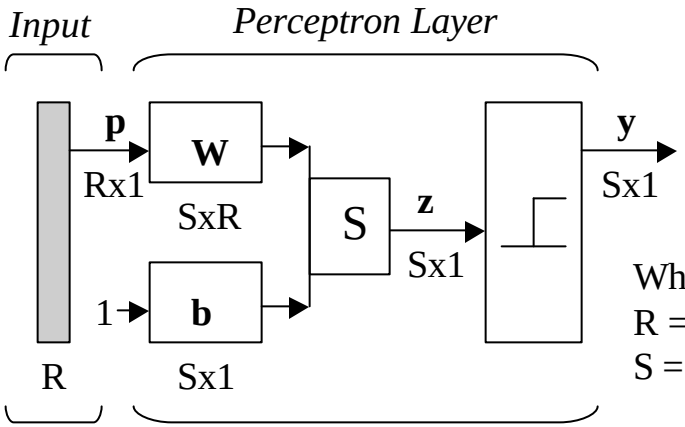
$$P = \begin{bmatrix} -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1; \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1; \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad T = [0\ 1\ 0\ 0\ 1\ 1\ 1\ 0]$$





One-layer multi-perceptron classification of linearly separable patterns

MATLAB representation:



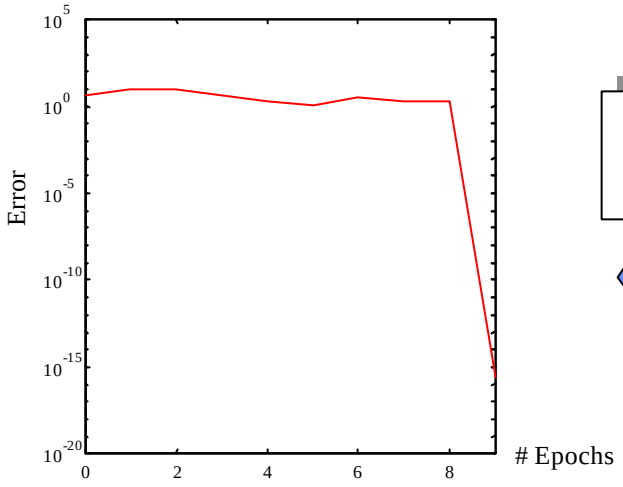
y = hardlim(**W***p+**b**)

Demo P3 in the "MATLAB Neural Network Toolbox - User's Guide"

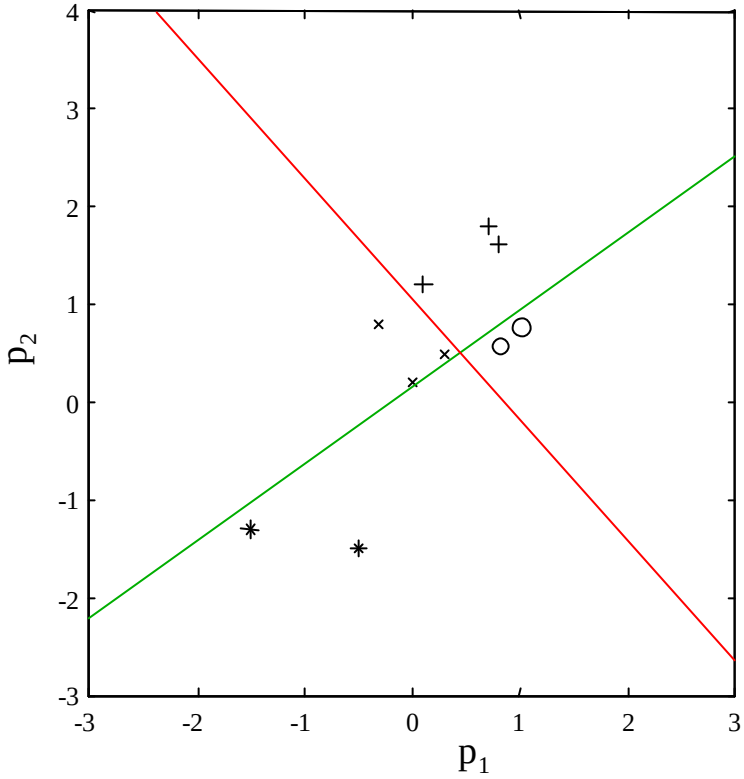
P = [0.1 0.7 0.8 0.8 1.0 0.3 0.0 -0.3 -0.5 -1.5;
1.2 1.8 1.6 0.6 0.8 0.5 0.2 0.8 -1.5 -1.3]

T = [1 1 1 0 0 1 1 1 0 0;
0 0 0 0 0 1 1 1 1 1]

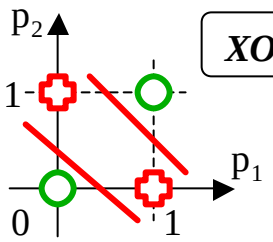
{ 00 = O ; 10 = + ; 01 = * ; 11 = x }



R = 2 inputs
S = 2 neurons



Perceptron Networks for Linearly Non-Separable Vectors

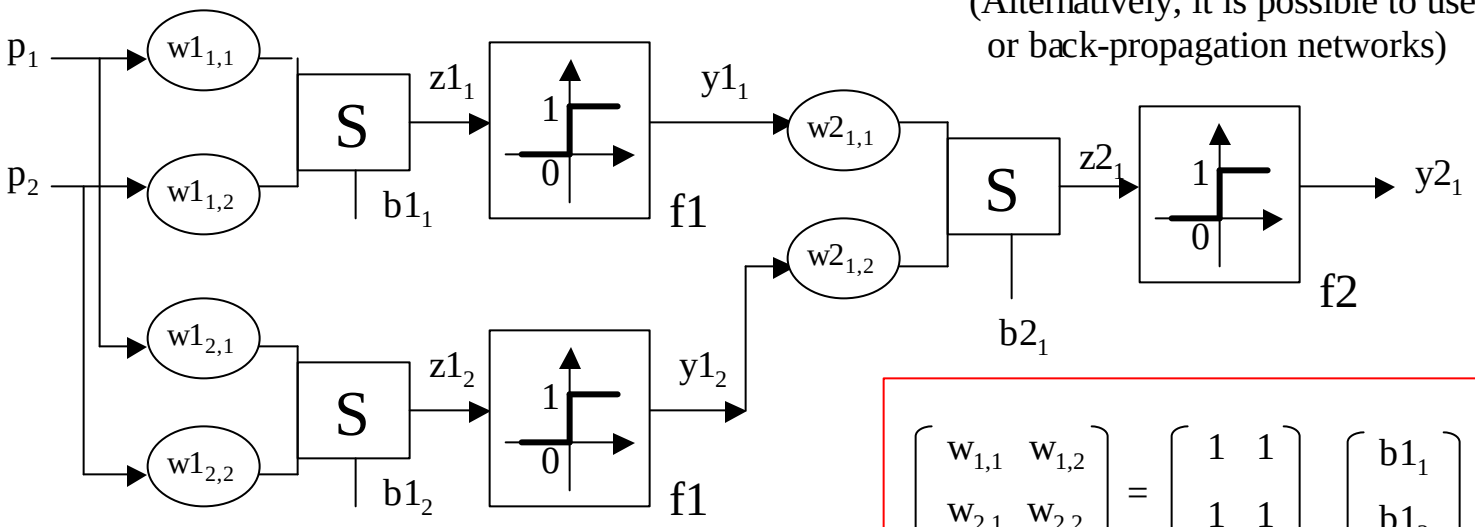


$\mathbf{p} = [\,0\ 0\ 1\ 1\,;\ 0\ 1\ 0\ 1\,]$

$\mathbf{t}_{XOR} = [\,0\ 1\ 1\ 0\,]$

If a straight line cannot be drawn between the set of input vectors associated with targets of 0 value and the input vectors associated with targets of 1, than a perceptron cannot classify these input vectors.

One solution is to use a two layer architecture, the perceptrons in the first layer are used as preprocessors producing linearly separable vectors for the second layer. (Alternatively, it is possible to use linear ANN or back-propagation networks)

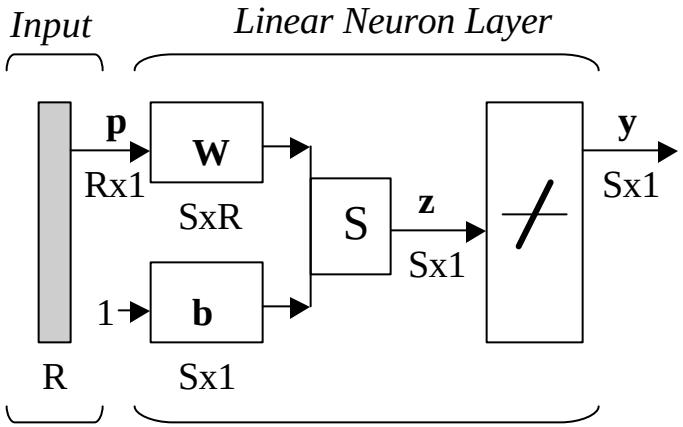


$$\begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad \begin{pmatrix} b1_1 \\ b1_2 \end{pmatrix} = \begin{pmatrix} -1.5 \\ -0.5 \end{pmatrix}$$

$[\,w2_{1,1}\ \ w2_{1,2}\,] = [-1\ \ 1]$ $\qquad$ $[\,b2_1\,] = [-0.5]$

The row index of a weight indicates the destination neuron of the weight and the column index indicates which source is the input for that weight.

LINEAR NEURAL NETWORKS (*ADALINE* NETWORKS)

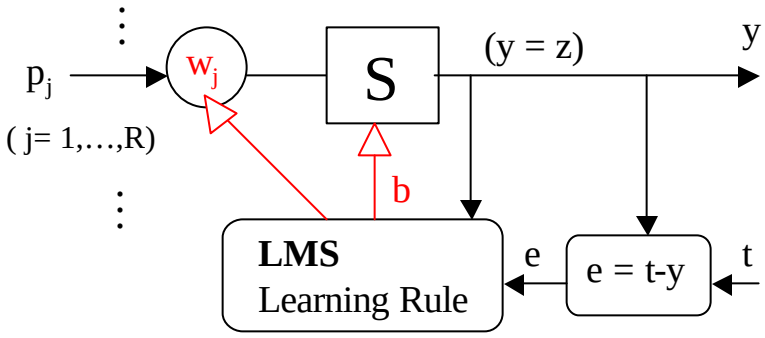


- Linear neurons have a *linear transfer function* that allows to use a Least Mean-Square (LMS) procedure - *Widrow-Hoff learning rule*- to adjust weights and biases according to the magnitude of errors.
- Linear neurons suffer from the same limitation as the perceptron networks: they can only solve *linearly separable problems.*

Where: R = # Inputs, S = # Neurons

(*ADALINE* <== *ADA*ptive *LI*near *NE*uron)

Widrow-Hoff Learning Rule (The) Rule)



The LMS algorithm will adjust ADALINE's weights and biases in such away to *minimize the mean-square-error E [e^2] between all sets of the desired response and network's actual response*:

$$E[(t-y)^2] = E[(t - (w_1 \dots w_R \; b)\cdot(p_1 \dots p_R \; 1)^T)^2] = E[(t - \mathbf{W}\cdot\mathbf{p})^2]$$

(NB: E[…] denotes the "expected value"; **p** is column vector)

$E[\,e^2\,] = E[\,(t - \mathbf{W}\cdot\mathbf{p})^2\,] = \{$as for deterministic signals the expectation becomes a time-average$\}$

$= E[t^2] - 2\cdot\mathbf{W}\cdot E[t\cdot\mathbf{p}] + \mathbf{W}\cdot E[\mathbf{p}\cdot\mathbf{p}^T]\cdot\mathbf{W}^T$

The cross-correlation between the input vector and its associated target.

The input cross-correlation matrix

If the **input correlation matrix is positive** the LMS algorithm will converge as there will a *unique minimum of the mean square error.*

- The W-H rule is an iterative algorithm uses the "steepest-descent" method to reduce the mean-square-error. The key point of the W-H algorithm is that it replaces $E[e^2]$ estimation by the squared error of the iteration *k*: $e^2(k)$. At each iteration step *k* it estimates the gradient of this error ∇_k with respect to **W** as a vector consisting of the partial derivatives of $e^2(k)$ with respect to each weight:

$$\nabla_k^* = \frac{\partial e^2(k)}{\partial W(k)} = \left[\frac{\partial e^2(k)}{\partial w_1(k)} \cdots \frac{\partial e^2(k)}{\partial w_R(k)}, \frac{\partial e^2(k)}{\partial b(k)}\right]$$

The weight vector is then modified in the direction that decreases the error:

$$W(k+1) = W(K) - m \bullet \nabla_k^* = W(k) - m \bullet \frac{\partial e^2(k)}{\partial W(k)} = W(k) - 2m \bullet e(k) \bullet \frac{\partial e(k)}{\partial W(k)}$$

- As t(k) and p(k) - both affecting e(k) - are independent of **W**(k), we obtain the final expression of the **Widrow-Hoff learning rule**:

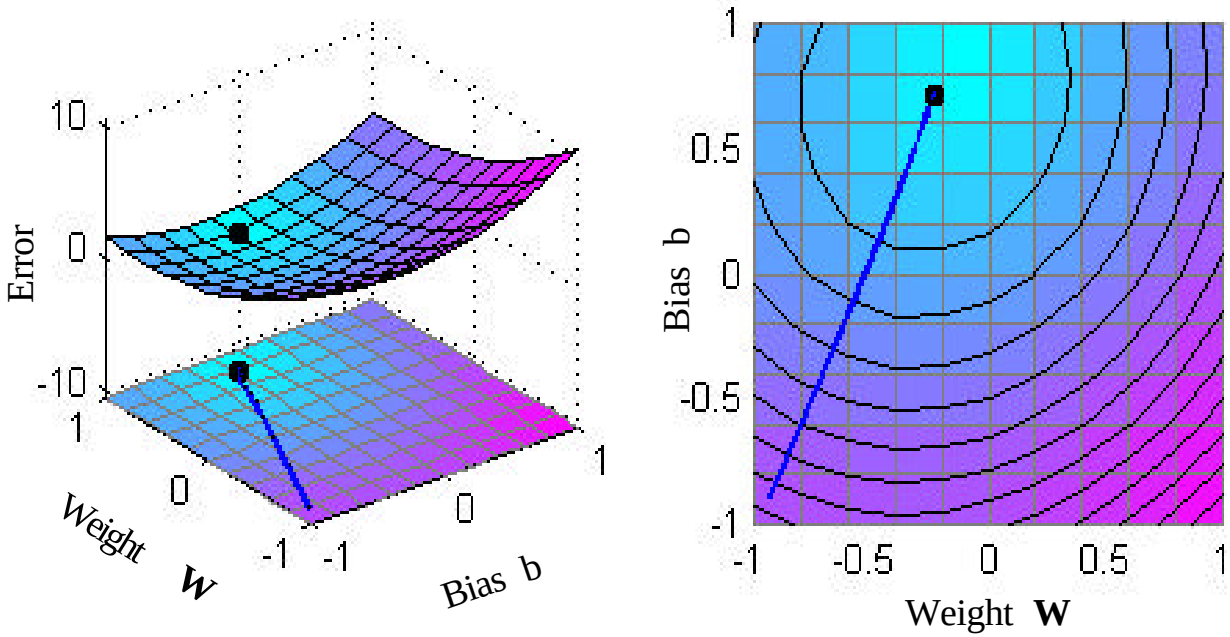
$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\mu\cdot e(k)\cdot\mathbf{p}(k) \quad \Rightarrow \quad b(k+1) = b(k) + 2\mu\cdot e(k)$$

where *m* the "learning rate" and $e(k) = t(k) - y(k) = t(k) - \mathbf{W}(k)\cdot\mathbf{p}(k)$

Demo Lin 2 in the "MATLAB Neural Network Toolbox - User's Guide"

P = [1.0 -1.2]
T = [0.5 1.0]

One-neuron one-input ADALINE, starting from some random values for w = -0.96 and b= -0.90 and using the "*trainwh*" MATLAB NN toolbox function, reaches the target after 12 epochs with an error e < 0.001. The solution found for the weight and bias is:
w = -0.2354 and b= 0.7066.



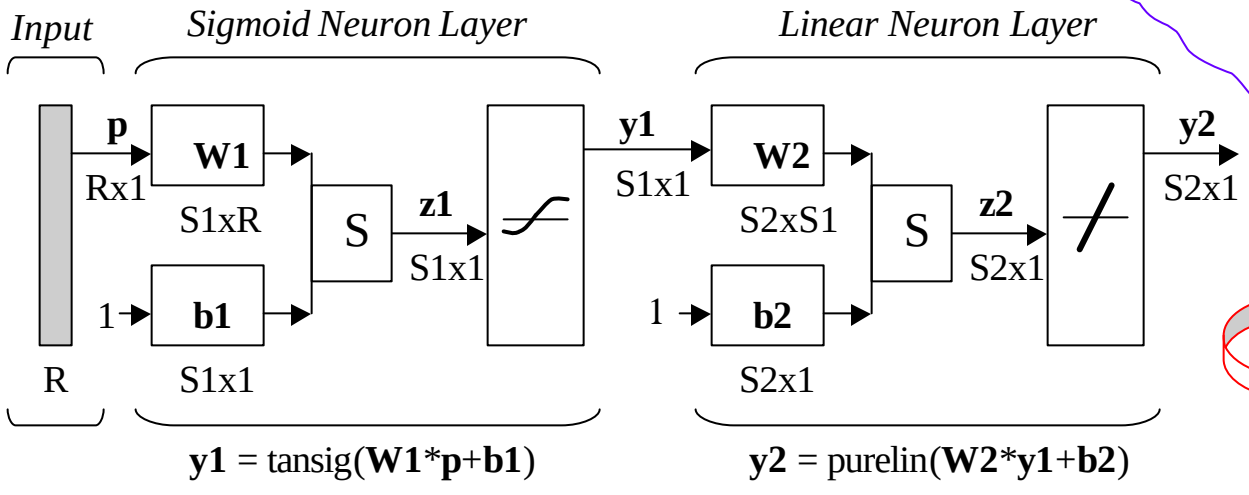
Back-Propagation Learning - The Generalized) Rule

P. Werbos (Ph.D. thesis 1974);
D. Parker (1985), Yann Le Cun(1985),
D. Rumelhart, G. Hinton, R. Williams (1986)

- Single layer ANNs are suitable to only solving linearly separable classification problems. Multiple feed-forward layers can give an ANN greater freedom. Any reasonable function can be modeled by a two layer architecture: a sigmoid layer feeding a linear output layer.
- Single layer ANNs are only able to solve linearly Widrow-Hoff learning applies to single layer networks. ==> generalized W-H algorithm (Δ -rule) ==> **back-propagation learning**.

- Back-propagation ANNs often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons.

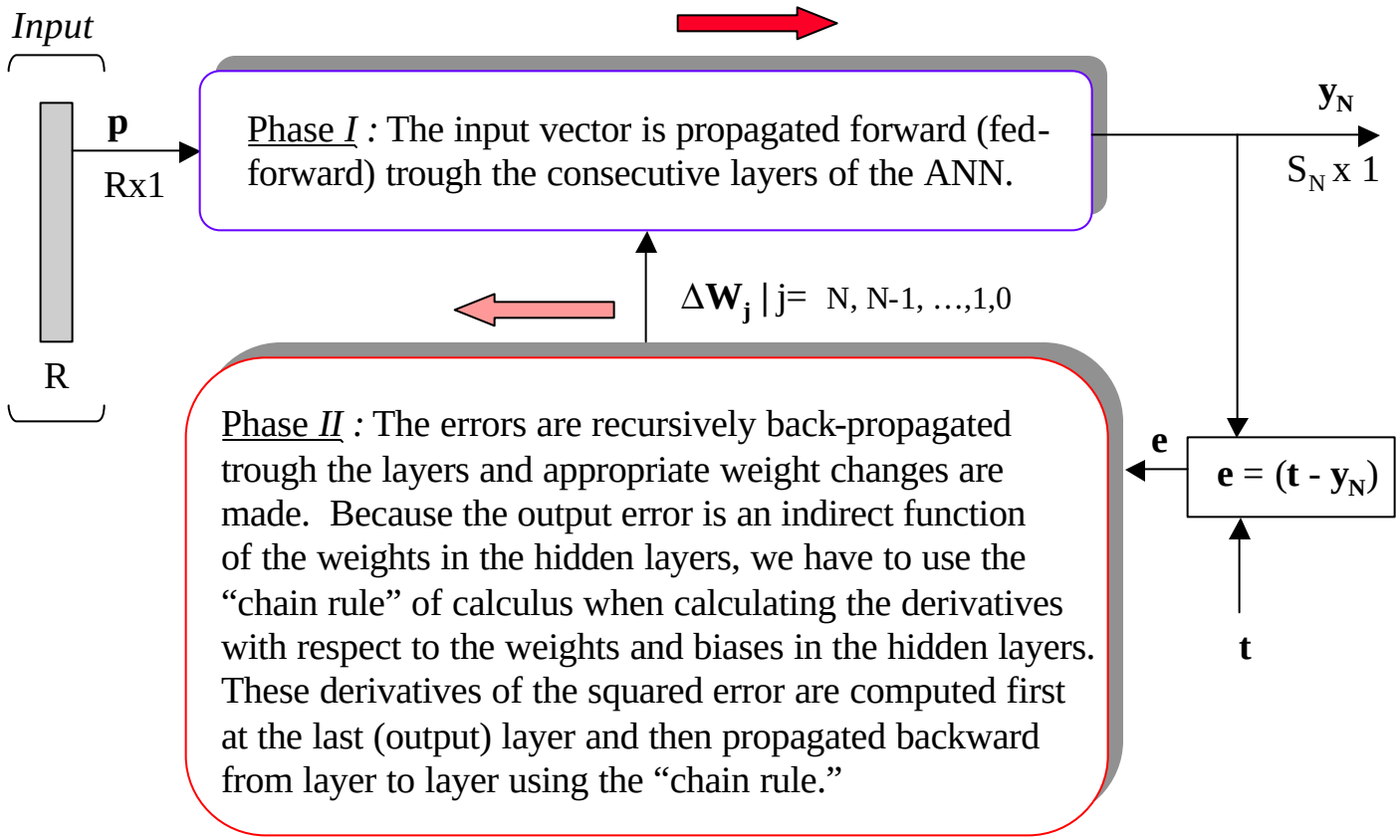
Two-layer ANN that can approximate any function with a finite number of of discontinuities, arbitrarily well, given sufficient neurons in the hidden layer.



e2= (t-y2) = (t- purelin (**W2***tansig(**W1*p** +**b1**) +**b2**))

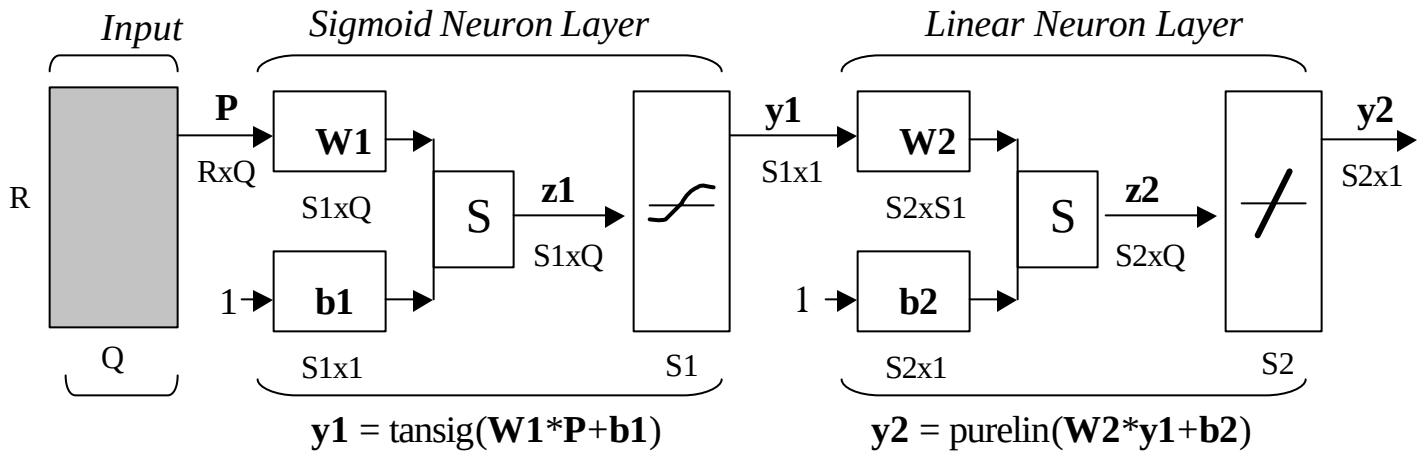
The error is an indirect function of the weights in the hidden layers.

- Back-propagation is an iterative steepest descent algorithm, in which the performance index is the mean square error E [e^2] between the desired response and network's actual response:

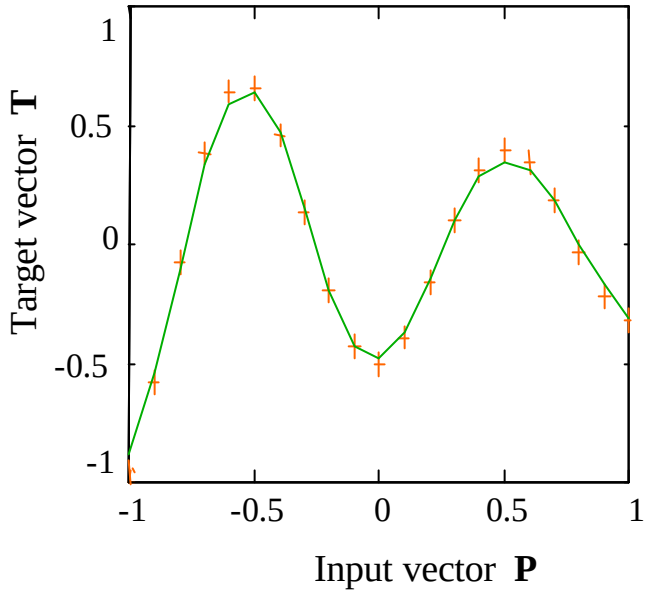


EXAMPLE: **Function Approximation by Back-Propagation**

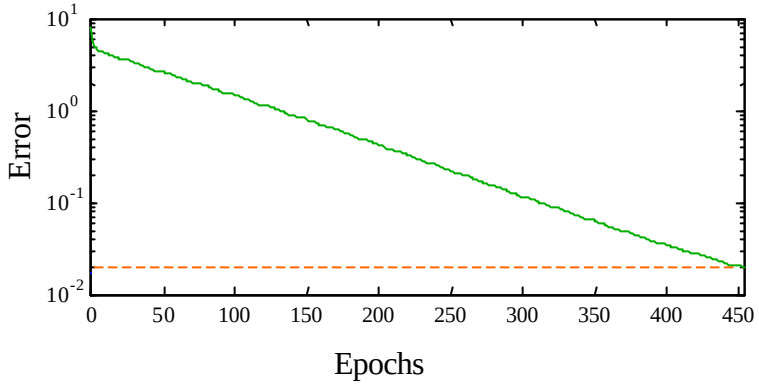
Demo BP4 in the" MATLAB Neural Network Toolbox User's Guide"



R = 1 input
S1 = 5 neurons in layer #1
S2 = 1 neuron in layer #2
Q = 21 input vectors



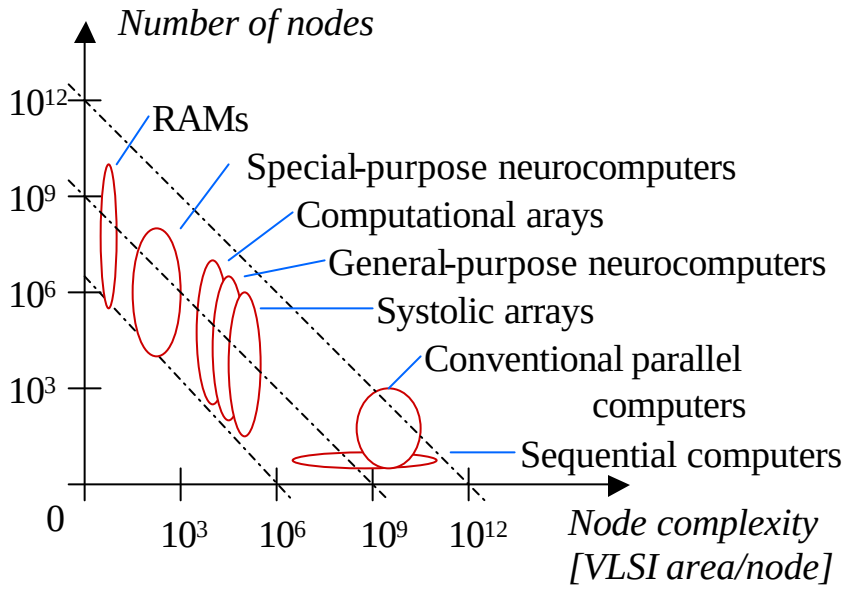
The back-propagation algorithm took 454 epochs to approximate the 21 target vectors with an error < 0.02



HARDWARE NEURAL NETWORK ARCHITECTUCTURES

ANNs / **Neurocomputers** ==>architectures optimized for neuron model implementation

- *general-purpose*, able to emulate a wide range of NN models;
- *special-purpose*, dedicated to a specific NN model.



[from P. Treleaven, M. Pacheco, M. Vellasco, "VLSI Architectures for Neural Networks," IEEE Micro, Dec. 1989, pp. 8-27]

ANN VLSI Architectures:

- *analog* ==> compact,high speed, asynchronous, no quantization errors, convenient weight "+"and "X";
- digital ==> more efifcient VLSI technology, robust, convenient weight storage;

Pulse Data Representation:

- *Pulse Amplitude Modulation* (PAM) - not satisfactory for NN processing;
- Pulse Width Modulation (PWM);
- Pulse Frequency Modulation (PFM).

Pulse Stream ANNs: combination of different pulse data representation methods and opportunistic use of both analog and digital implementation techniques.

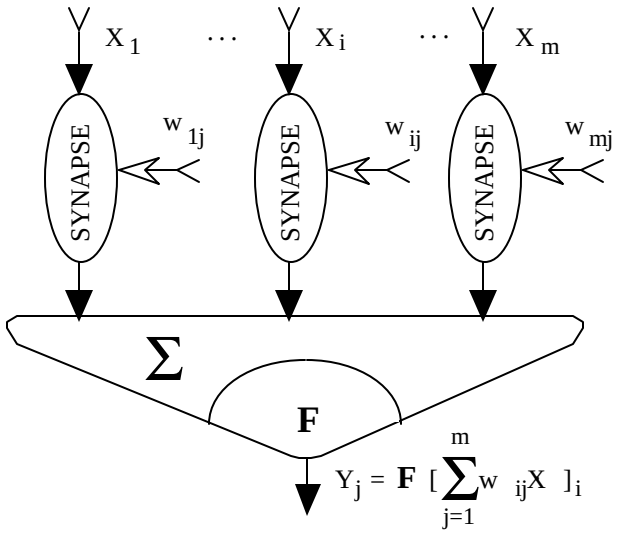
HARDWARE NEURAL NETWORK ARCHITECTURES USING RANDOM-PULSE DATA REPRESENTATION

- Looking for a model to prove that algebraic operations with analog variables can be performed by logical gates, **von Neuman** advanced in 1956 the idea of representing analog variables by the mean rate of random-pulse streams [*J. von Neuman, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in Automata Studies, (C.E. Shannon, Ed.), Princeton, NJ, Princeton University Press, 1956*].

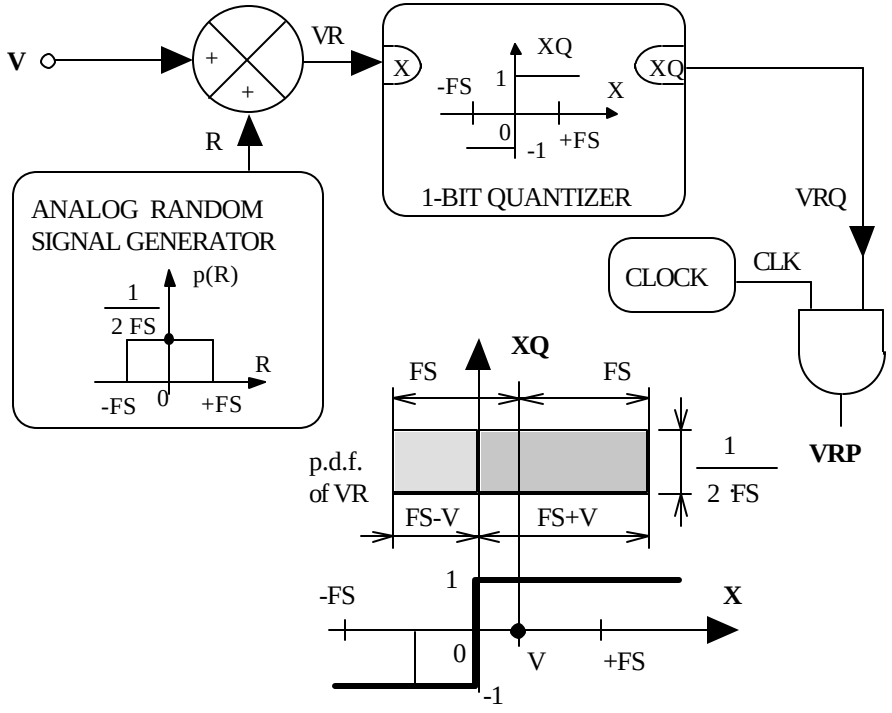
- The "**random-pulse machine**" concept, [*S.T. Ribeiro, "Random-pulse machines," IEEE Trans. Electron. Comp., vol. EC-16, no. 3, pp. 261-276,1967*], a.k.a. "noise computer", "stochastic computing", "dithering" deals with analog variables represented by the mean rate of random-pulse streams allowing to use digital circuits to perform arithmetic operations. This concept presents a good tradeoff between the electronic circuit complexity and the computational accuracy. The resulting neural network architecture has a high packing density and is well suited for very large scale integration (VLSI).

❖ HARDWARE *ANN* USING RANDOM-PULSE (1-BIT) DATA REPRESENTATION

[E.M. Petriu , K. Watanabe, T. Yeap, "Applications of Random-Pulse Machine Concept to Neural Network Design," IEEE Trans. Instrum. Meas., Vol. 45, No.2, pp.665-669, 1996.]



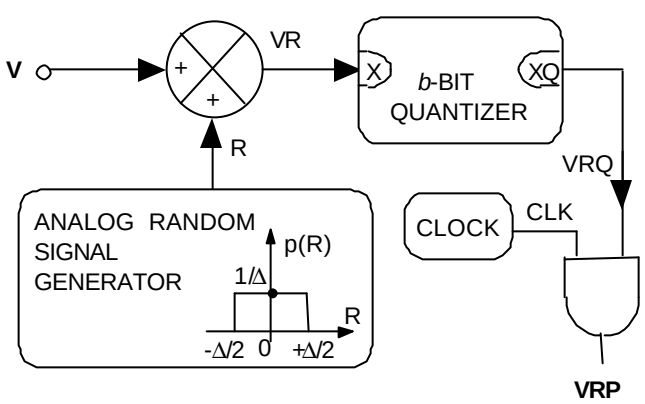
Neuron Structure

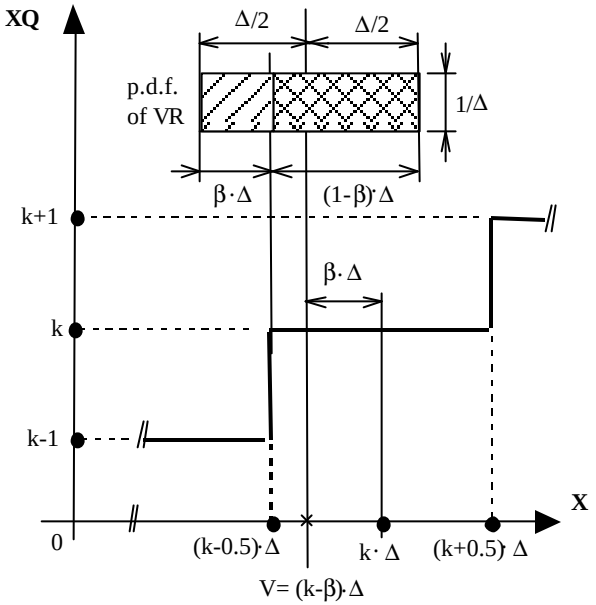


One-Bit "Analog / Random Pulse" Converter

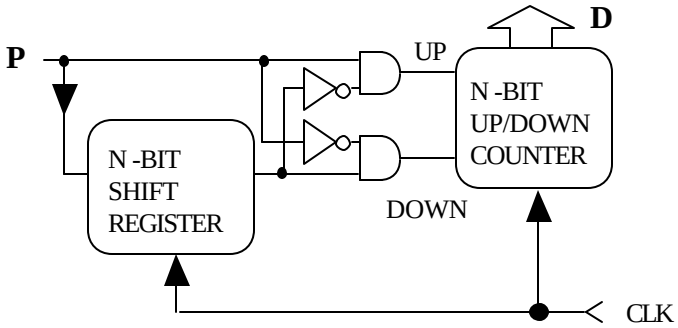
❖ HARDWARE *ANN* USING MULTI-BIT RANDOM-DATA REPRESENTATION

[E.M. Petriu , L. Zhao, S.R. Das, and A. Cornell, "Instrumentation Applications of Random-Data Representation," *Proc. IMTC/2000, IEEE Instrum. Meas. Technol. Conf.,* pp. 872-877, Baltimore, MD, May 2000]
[L. Zhao, "Random Pulse Artificial Neural Network Architecture," *M.A.Sc. Thesis, University of Ottawa,* 1998]

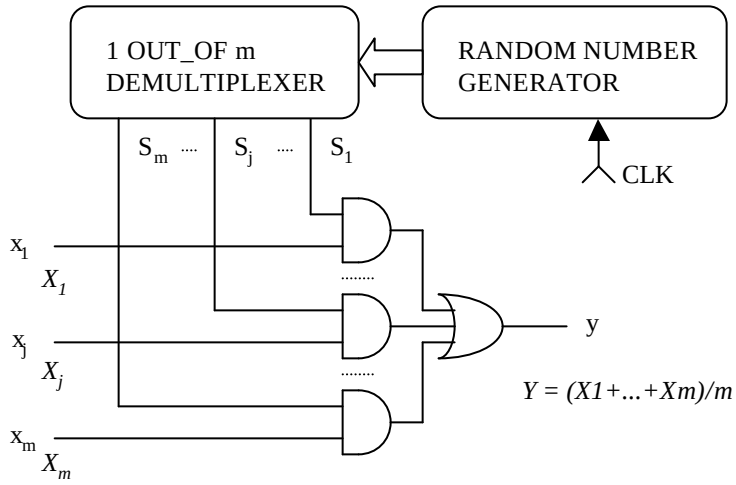




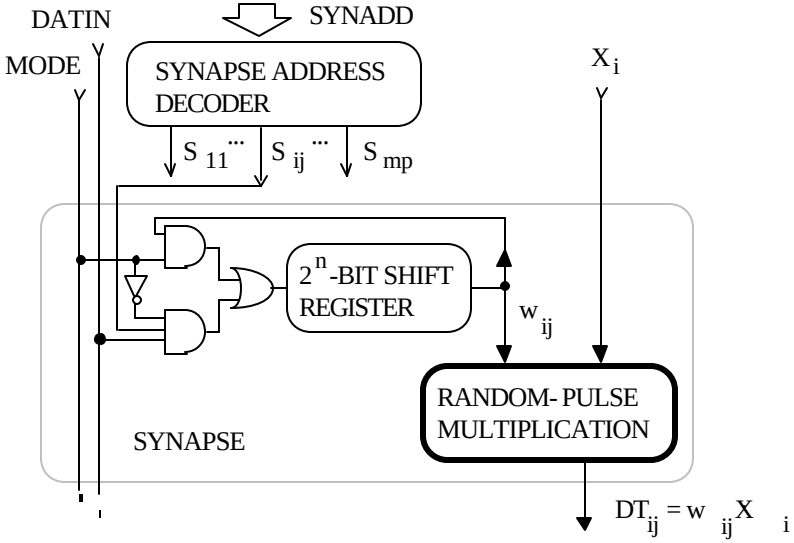
Generalized *b*-bit analog/random-data conversion and its quantization characteristics



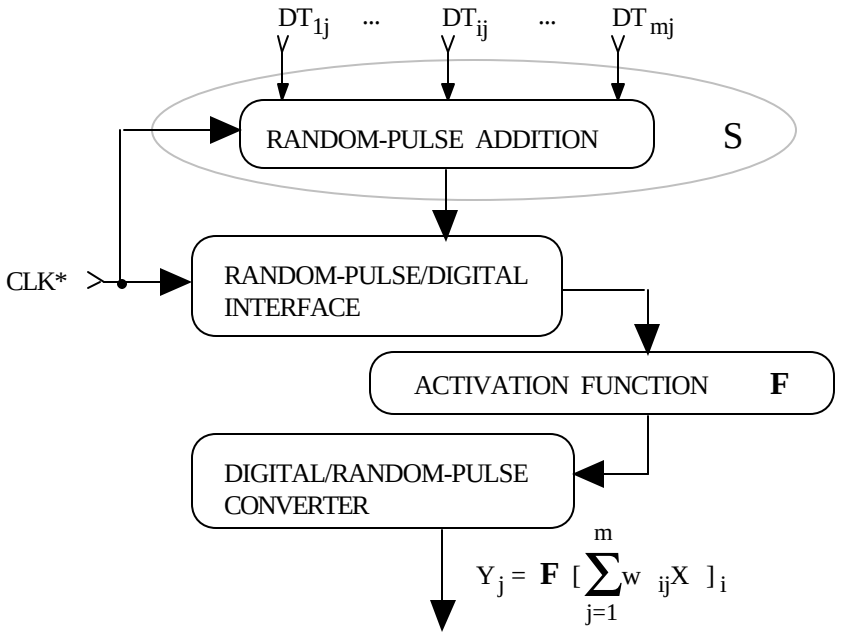
"Random Pulse / Digital" Converter
using a Moving Average Algorithm



Random Pulse Addition

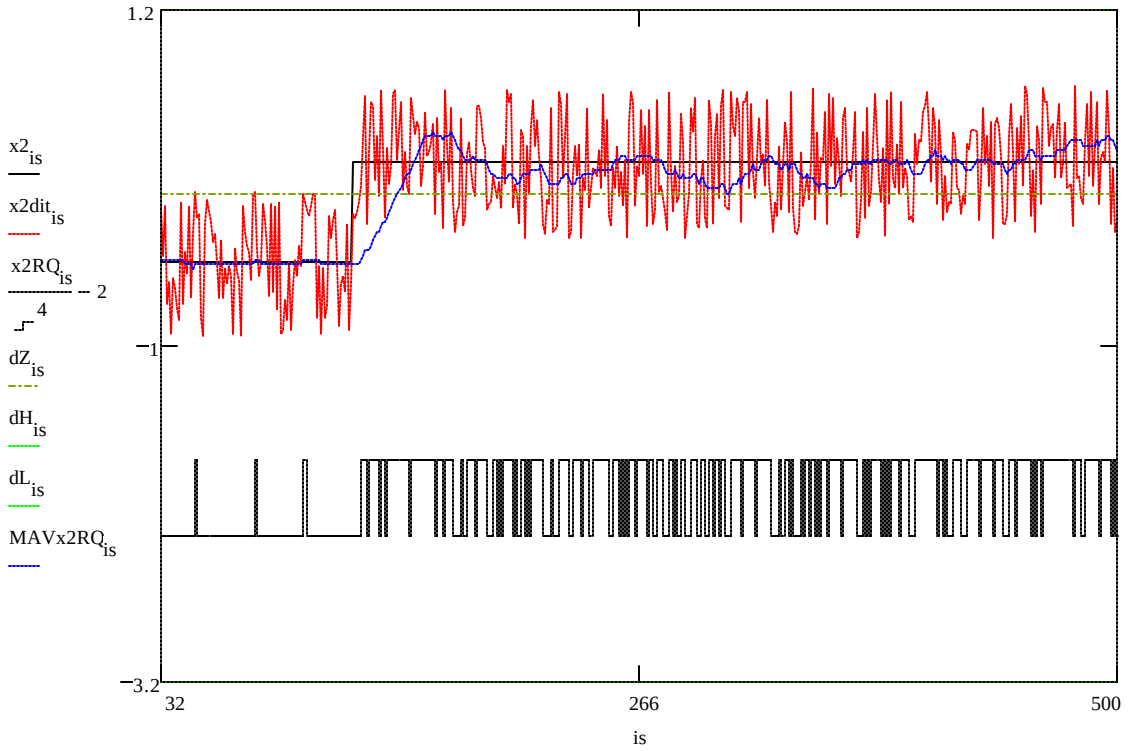


Random Pulse Implementation of a Synapse

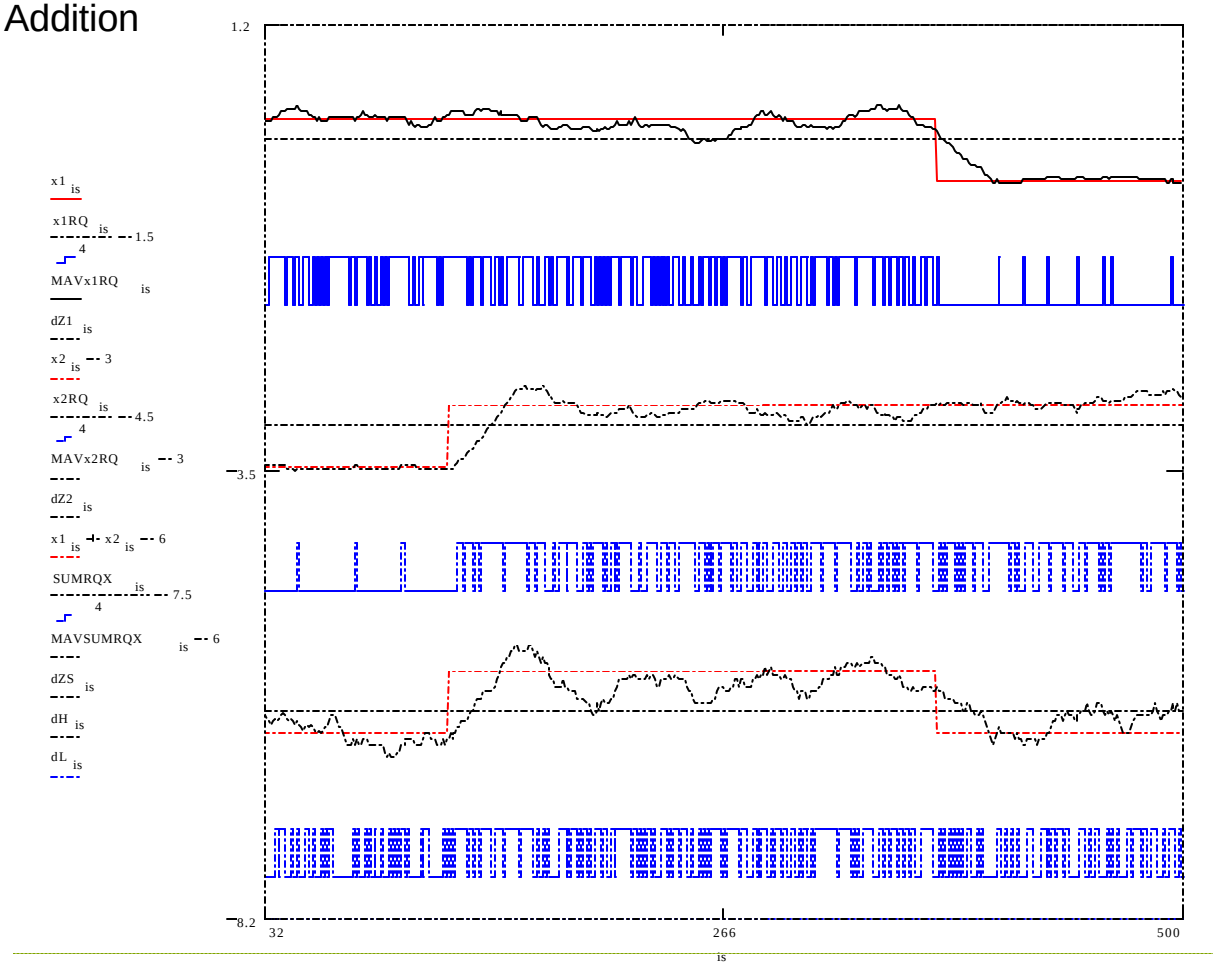


Neuron Body Structure

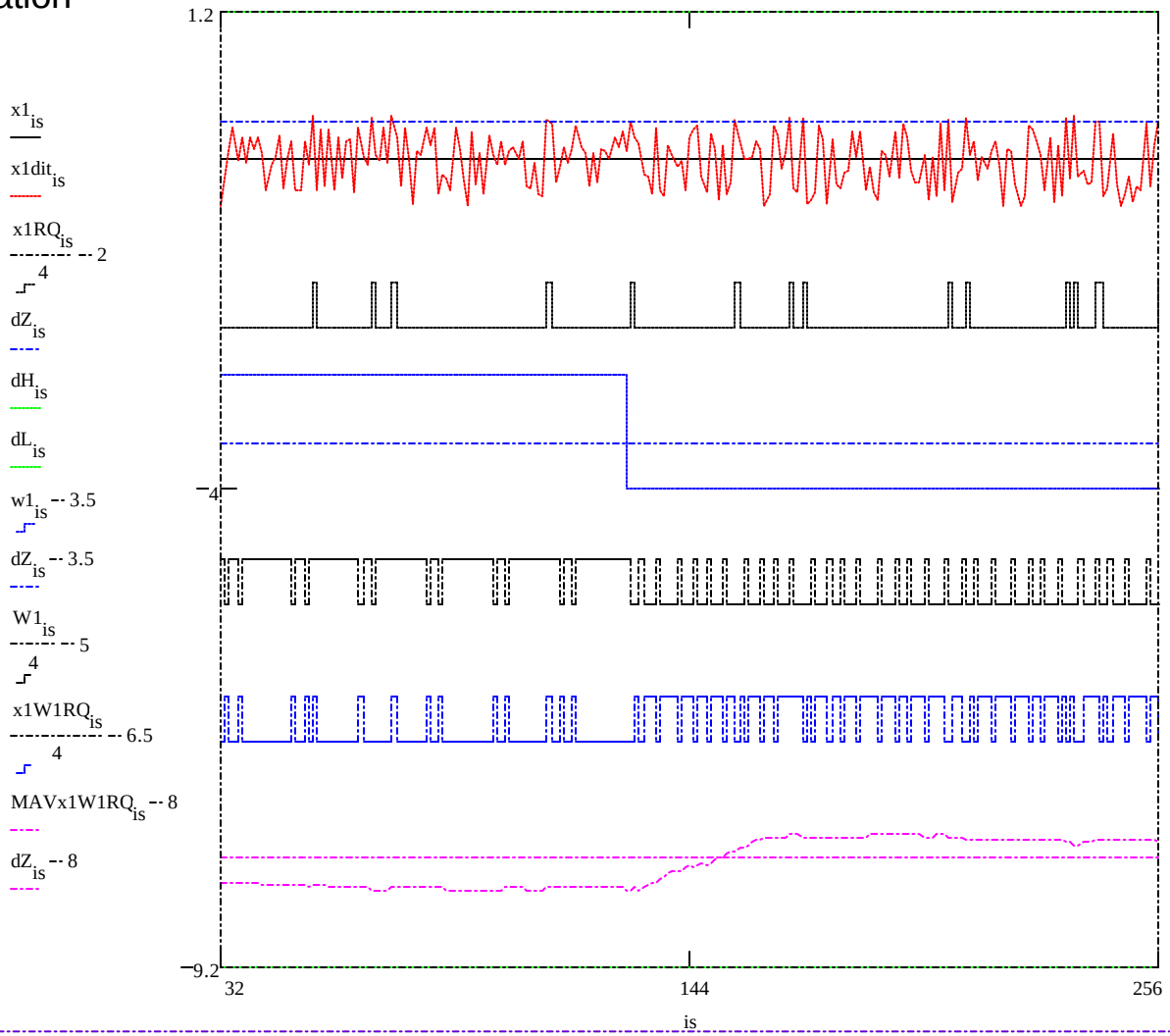
Moving Average 'Random Pulse -to- Digital " Conversion

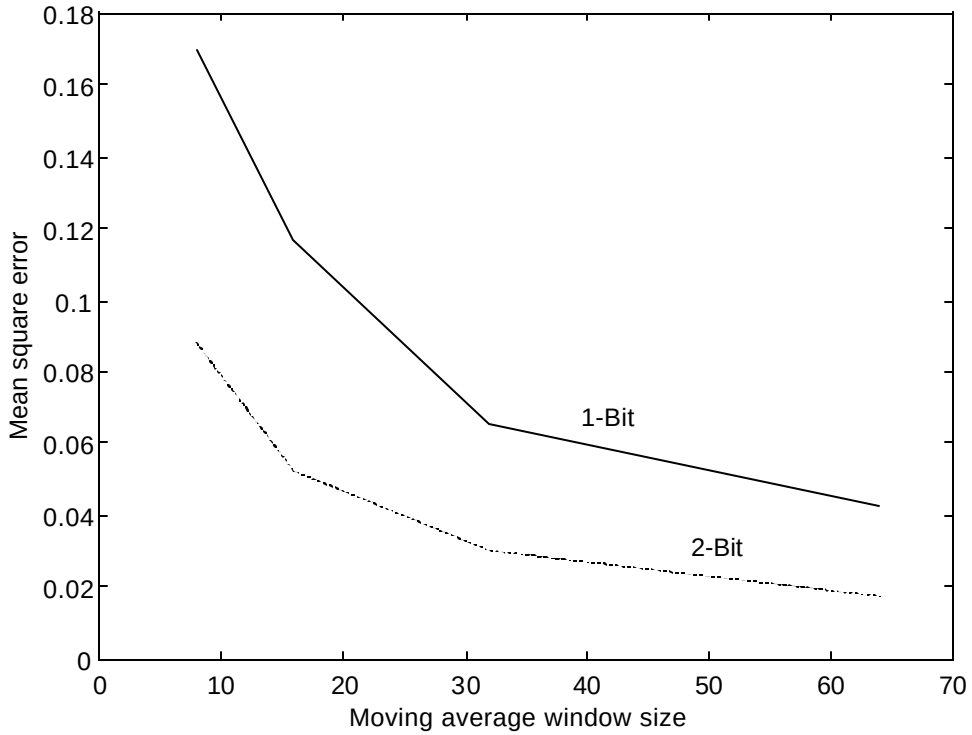


Random Pulse Addition

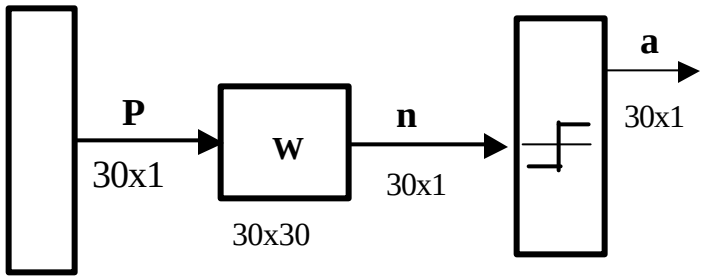


Random Pulse Multiplication



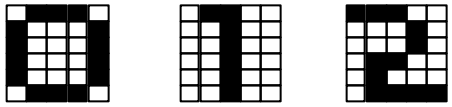


Mean square errors function of the moving average window size



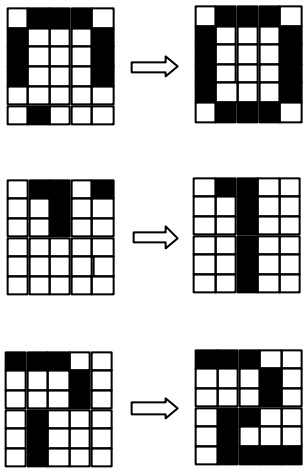
$a = \text{hardlim}(W * P)$

Auto-associative memory NN architecture



P_1, t_1 P_2, t_2 P_3, t_3

Training set for the auto-associative memory



Recovery of 30% occluded patterns

NEURAL NETWORK MODELS OF PHYSICAL PROCESSES

Modelling allows to simulate the behavior of a system for a variety of initial conditions,excitations and systems configurations - often in a much shorter time than would be required to physically build and test a prototype experimentally

- The ***quality and the degree of the approximation*** of the model can be determined only by a validation against experimental measurements.

- The ***convenience*** of the model means that it is capable of performing extensive parametric studies, in which independent parameters describing the model can be varied over a specified range in order to gain a global understanding of the response.

- A more relevant model might be one which provides results more rapidly - even if a degradation in a solution accuracy results.

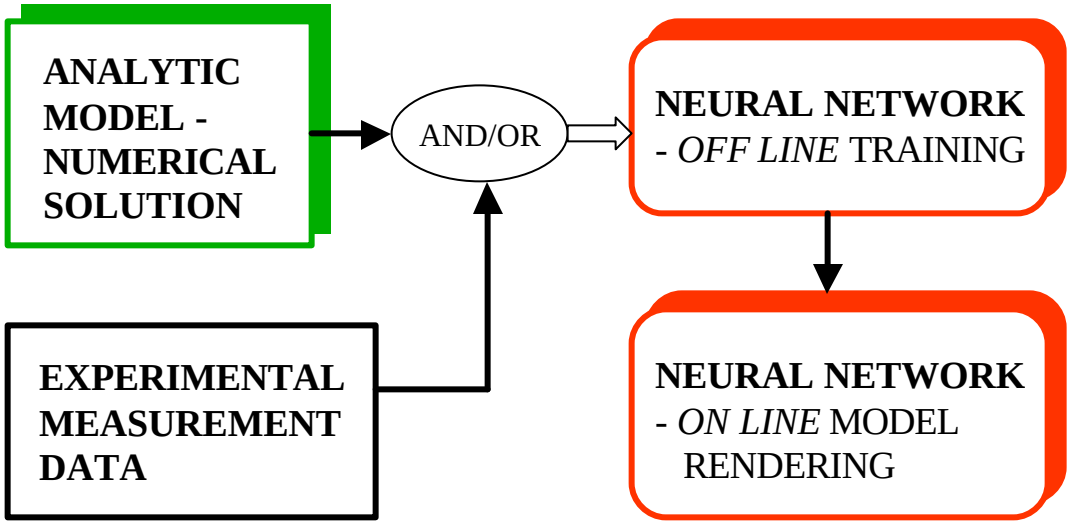
Analog Computer vs. Neural Network Tools for Physical Processes Modelling

- Both the Analog Computers and the Neural Networks are ***continuous modelling devices*.**

- The Analog Computer (AC) allows to *solve the linear or nonlinear differential and/or integral equations representing mathematical model* of a given physical process. The coefficients of these equations must be exactly known as they are used to program/adjust the coefficient-potentiometers of the AC's computing -elements (OpAmps). The AC doesn't follow a sequential computation, all its computing elements perform simultaneously and continuously.
  As an interesting note, "because of the difficulties inherent in analog differentiation the [differential] equation is rearranged so that it can be solved by integration rather than differentiation." [A.S. Jackson, *Analog Computation*, McGraw-Hill Book Co., 1960].

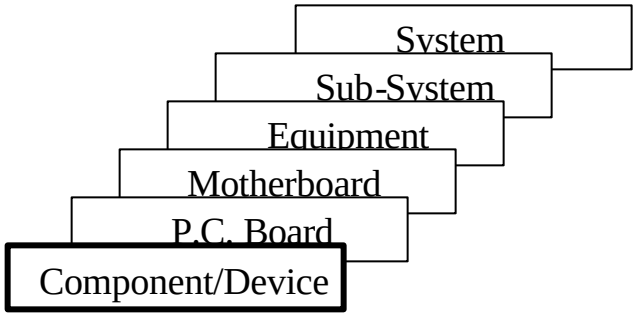
- The Neural Network (NN) doesn't require a prior mathematical model. A *learning algorithm* is used to adjust, sequentially by trail and error during the learning phase, the synaptic-weights/ coefficient-potentiometers of the neurons/computing-elements. As the AC, the NN don't follow a sequential computation, all its neuron performing simultaneously and continuously. The neurons are also integrative-type computing/processing elements.

Neural Network Models



EMC Modelling for Electronic Design Automation

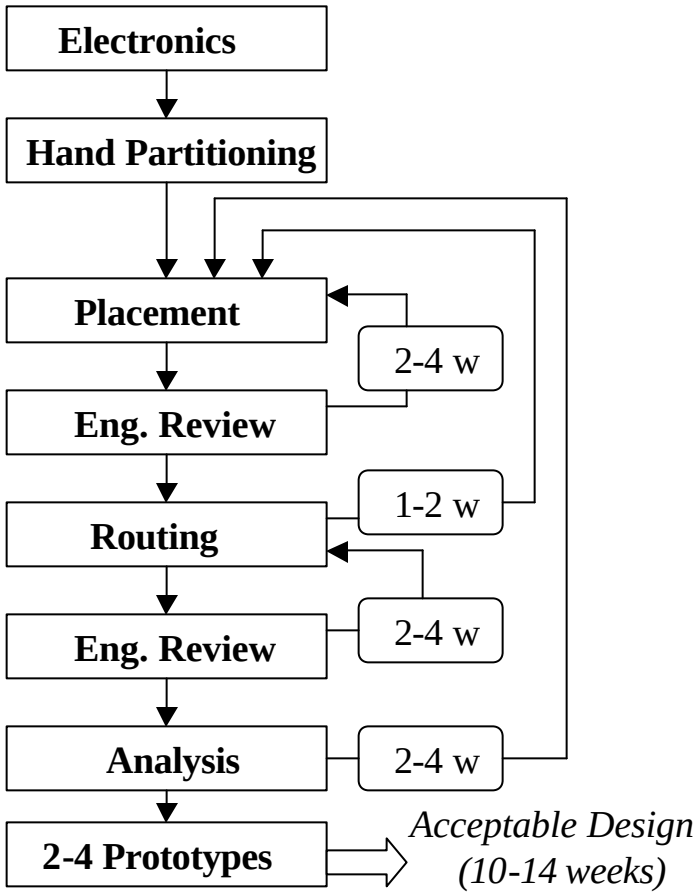
EMC Design Levels

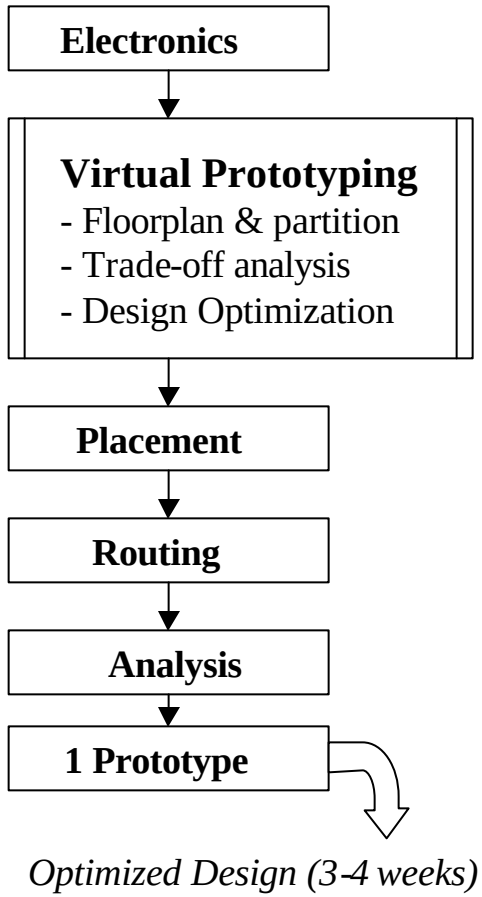


Optimum Approach to EMC Design

- {Design+Test+Analysis} **Synergy**
- **EMC_Behavior** = *F* (Design_Principle, Analysis&Modeling&Simulation_Tools, Test_Methodology&Instrumentation)

Product Design Cycles for Traditional and Virtual Prototyping





Electromagnetic Compatibility (EMC) Modelling Methods

- *circuit theory* to describe the conducted disturbances (such as overvoltages, voltage dips, voltage interruptions, harmonics, common ground coupling);

- *equivalent circuit* with either *distributed* or *lumped parameters* (such as in low frequency electromagnetic field coupling expressed in terms of mutual inductances and stray capacitances, field-to-line coupling using the transmission line approximation, and cable crosstalk);

- formal solutions to *Maxwell's equations* and the appropriate field boundary conditions (as for example in problems involving antenna scattering and radiation).

Electromagnetic Models Based on Maxwell's Equations

- ***linear phenomena***: appropriate models can be developed directly from Maxwell's equations;
- ***nonlinear behavior:*** necessary to combine measurements of the nonlinear behavior together with the Maxwell equations.

Parallel and Distributed Processing Techniques for Electromagnetic Field Solution

* **Classical numerical EM modelling** using sequential algorithms such as TLM (transmission-line matrix) or FEM (finite element method) is computer intensive, particularly as spatial discretization, geometry complexity, and domain size requirements become more demanding.

* More efficient **parallel and distributed computing** techniques must be developed to reduce the execution time for these methods so that they can be used in commercial CAD software. Speed of execution is particularly important when the field analysis is to be coupled with optimization, which may require several hundred analyses to be performed within a reasonable time. ⇒ **NN models**

NN modeling of the 3D EM field radiated by a dielectric-ring resonator antenna

[I. Ratner, H.O. Ali, E.M. Petriu, "Neural Network Simulation of a Dielectric Ring Resonator Antenna," J. Systems Architecture, vol. 44, No. 8, pp. 569-581, 1998.]

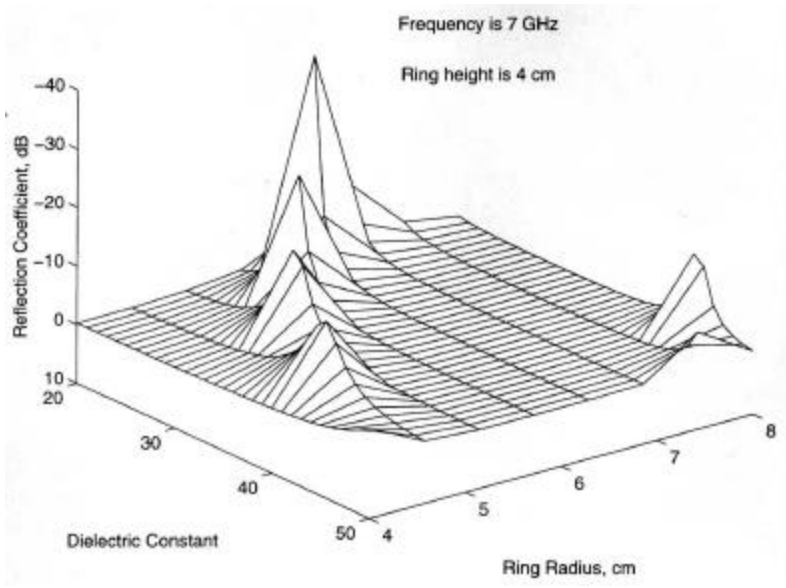
Maxwell's equations:

$$\nabla \times \overline{H} = (\sigma + j\omega\varepsilon)\overline{E}$$

$$\nabla \times \overline{E} = -j\omega\mu\overline{H}$$

$$\nabla \times \nabla \times \overline{H} = -j\omega\mu\,(\sigma + j\omega\varepsilon)\overline{H}$$

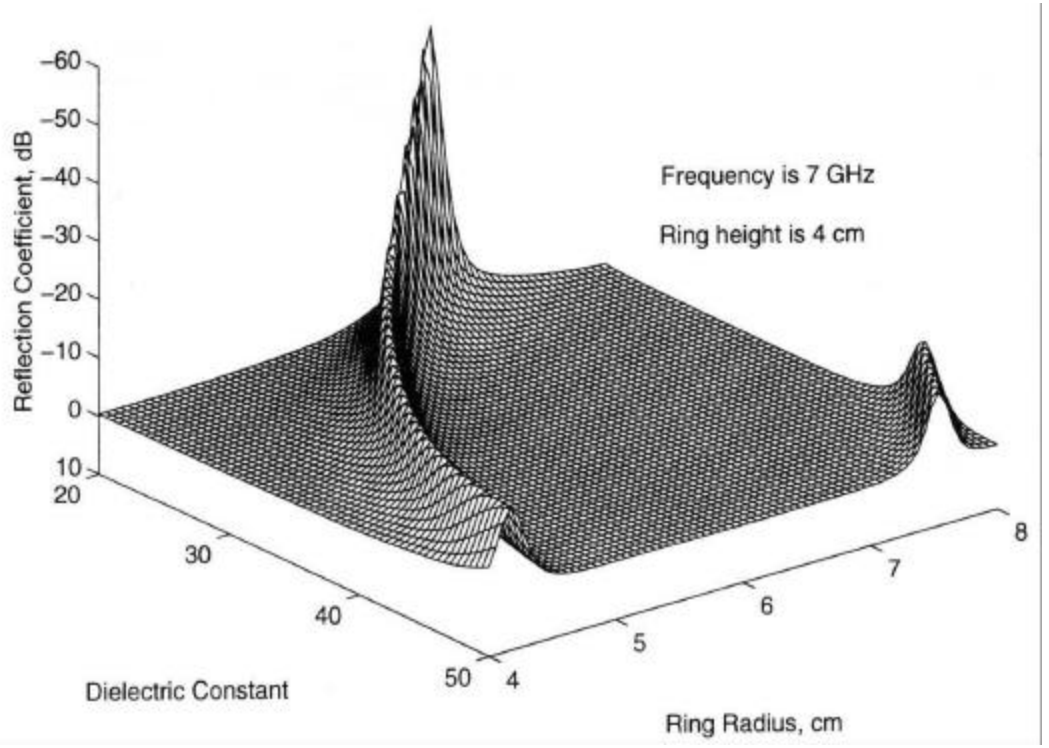
Finite Element Method (*FEM*)
1400 frequency steps 2-16 GHz;
31 dielectric constants; a = d = 5.14 mm



FEM numerical Solution => **1.3×10^5 s** on SPARC 10 UNIX

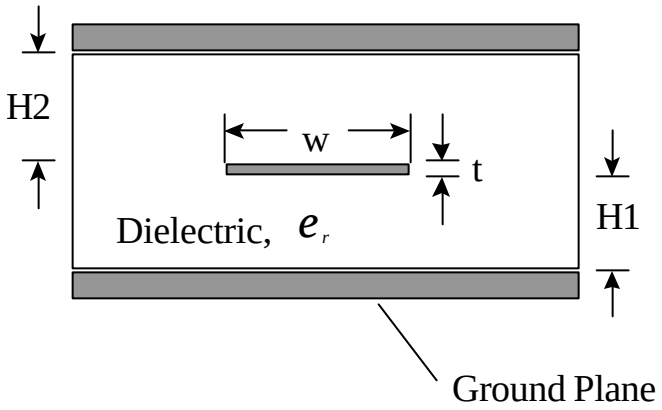
NEURAL NETWORK

- Two input neurons (frequency, dielectric constant) + Two hidden layers (5 neurons each, with hyperbolic tangent activation function) + One output linear neuron;
- Backpropagation using the Levenberg-Marquard algorithm;
- **55 s** /200 epochs to *train the NN off line* on SPARC 10 UNIX station;
- **0.5 s** to *render on line 5,000 points* of the EM field surface- model, SPARC 10 UNIX.



Modeling Single Stripline Interconnects

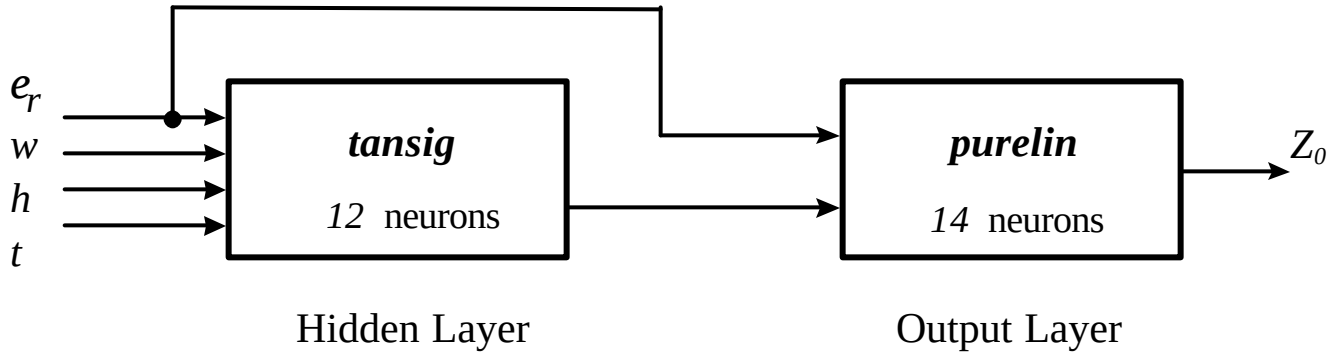
[Mao Jie, "NN Modeling of Single Stripline Interconnects," Technical Report, SMRLab, SITE, University of Ottawa, 1998

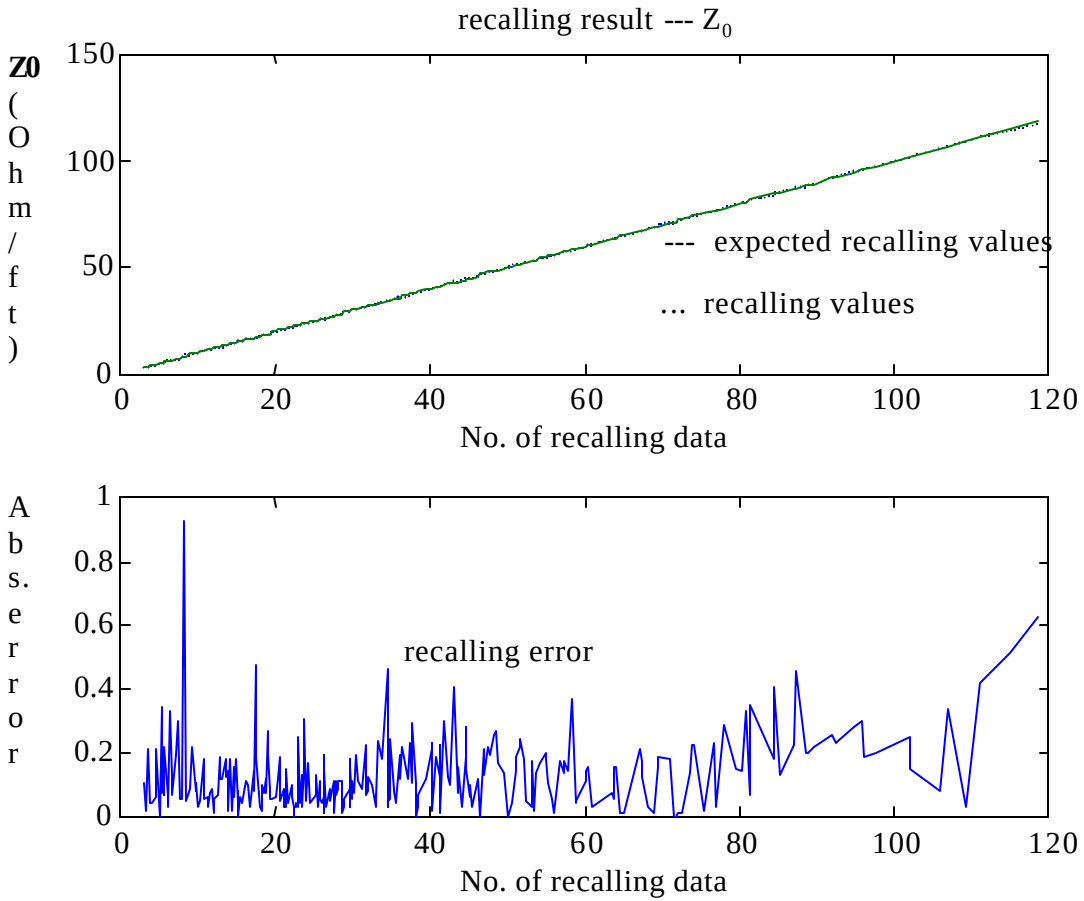


⇨ Model for $\mathbf{Z_0}$.

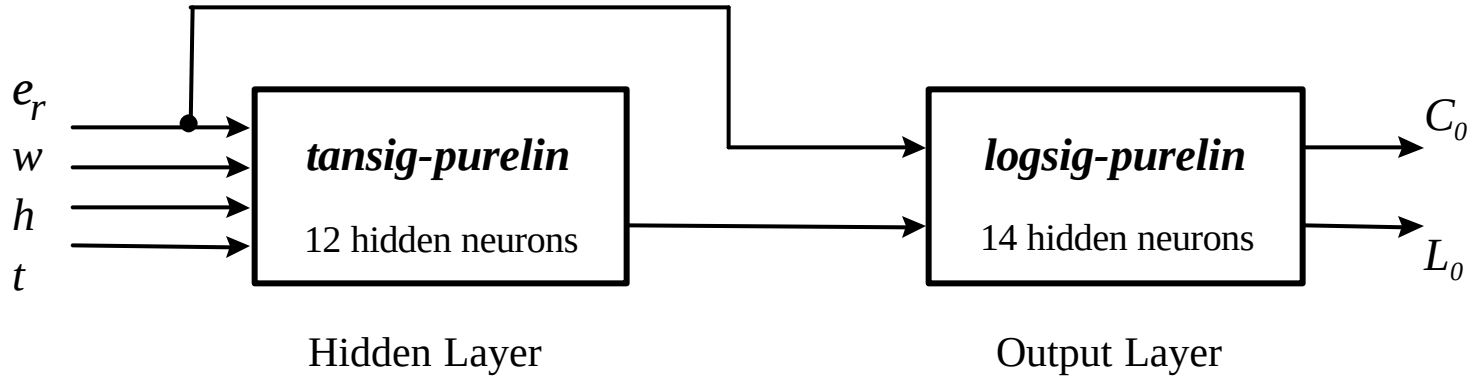
⇨ Model for $\mathbf{C_0}$ and $\mathbf{L_0}$.

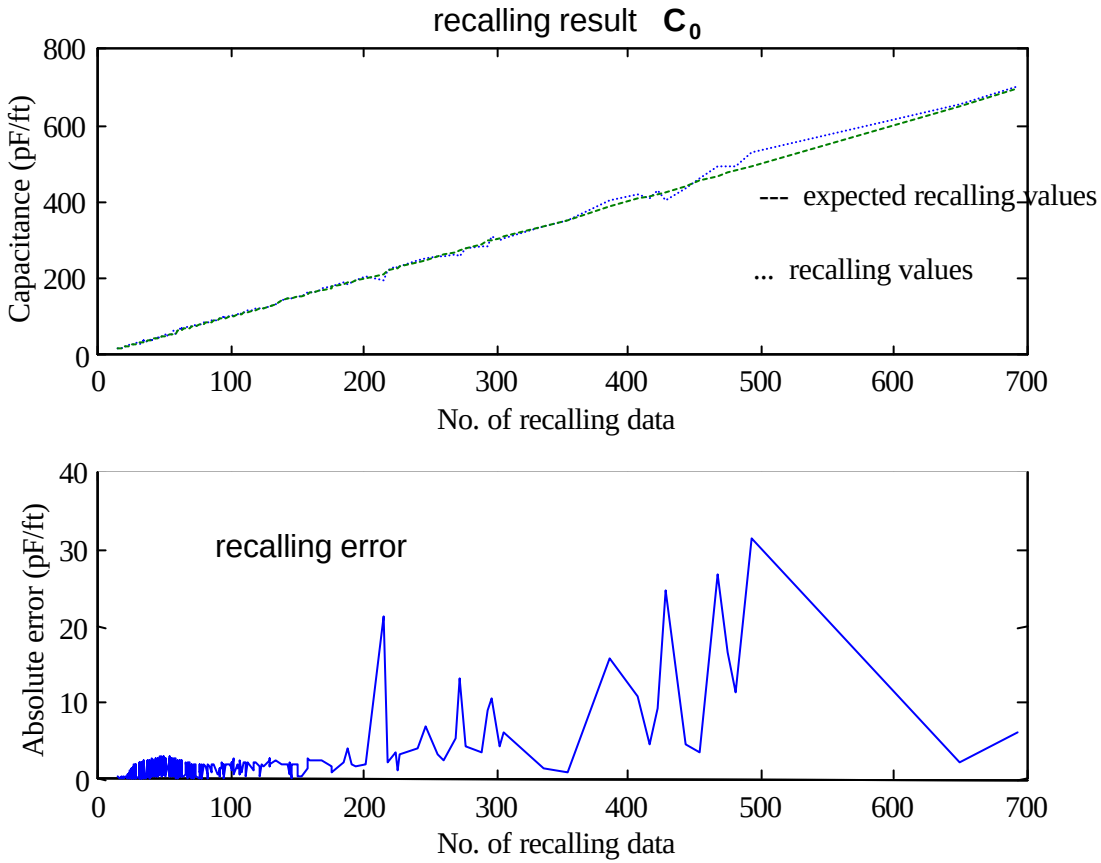
NN architecture modelling Z_0

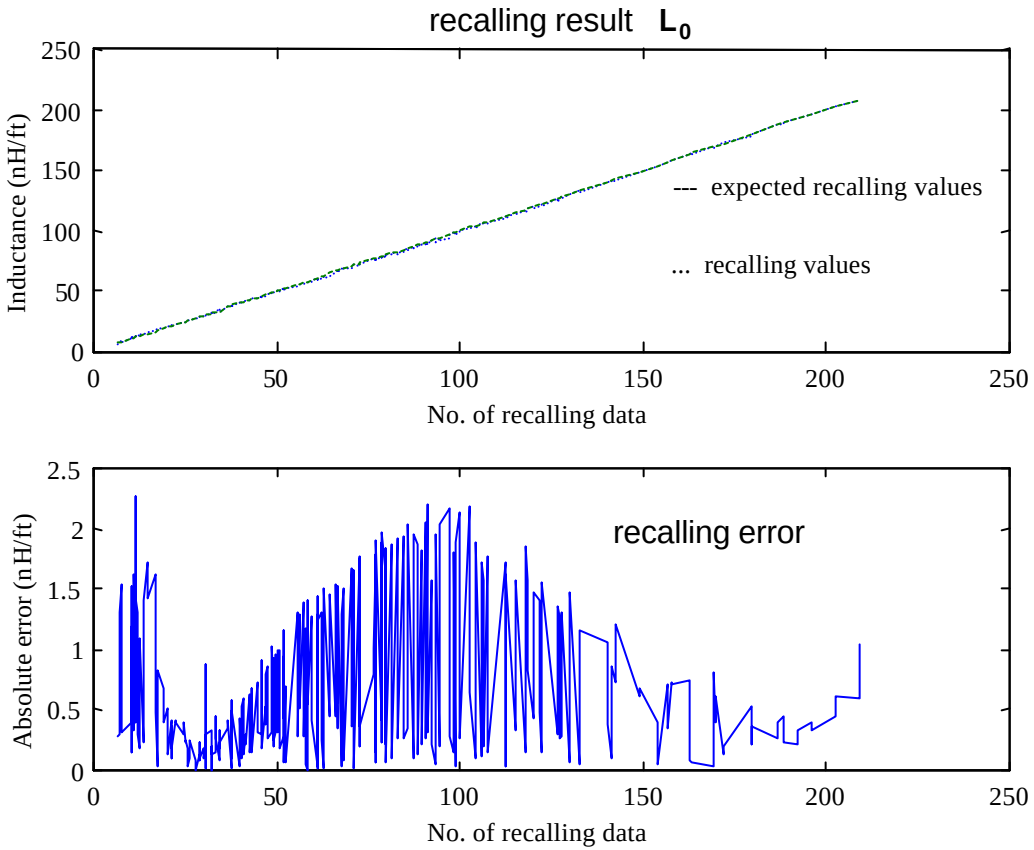



recalling result --- Z_0
Z0 (Ohm/ft)
150
100
50
0
0
20
40
60
80
100
120
No. of recalling data
--- expected recalling values
... recalling values
Abs. error
1
0.8
0.6
0.4
0.2
0
recalling error
No. of recalling data

NN architecture modelling C_0 and L_0




recalling result C_0
Capacitance (pF/ft)
No. of recalling data
--- expected recalling values
... recalling values
recalling error
Absolute error (pF/ft)
No. of recalling data


recalling result L_0
Inductance (nH/ft)
No. of recalling data
--- expected recalling values
... recalling values
recalling error
Absolute error (nH/ft)
No. of recalling data

NEURAL NETWORK MODELLING OF PLAIN AND GROOVED MICROSTRIPS

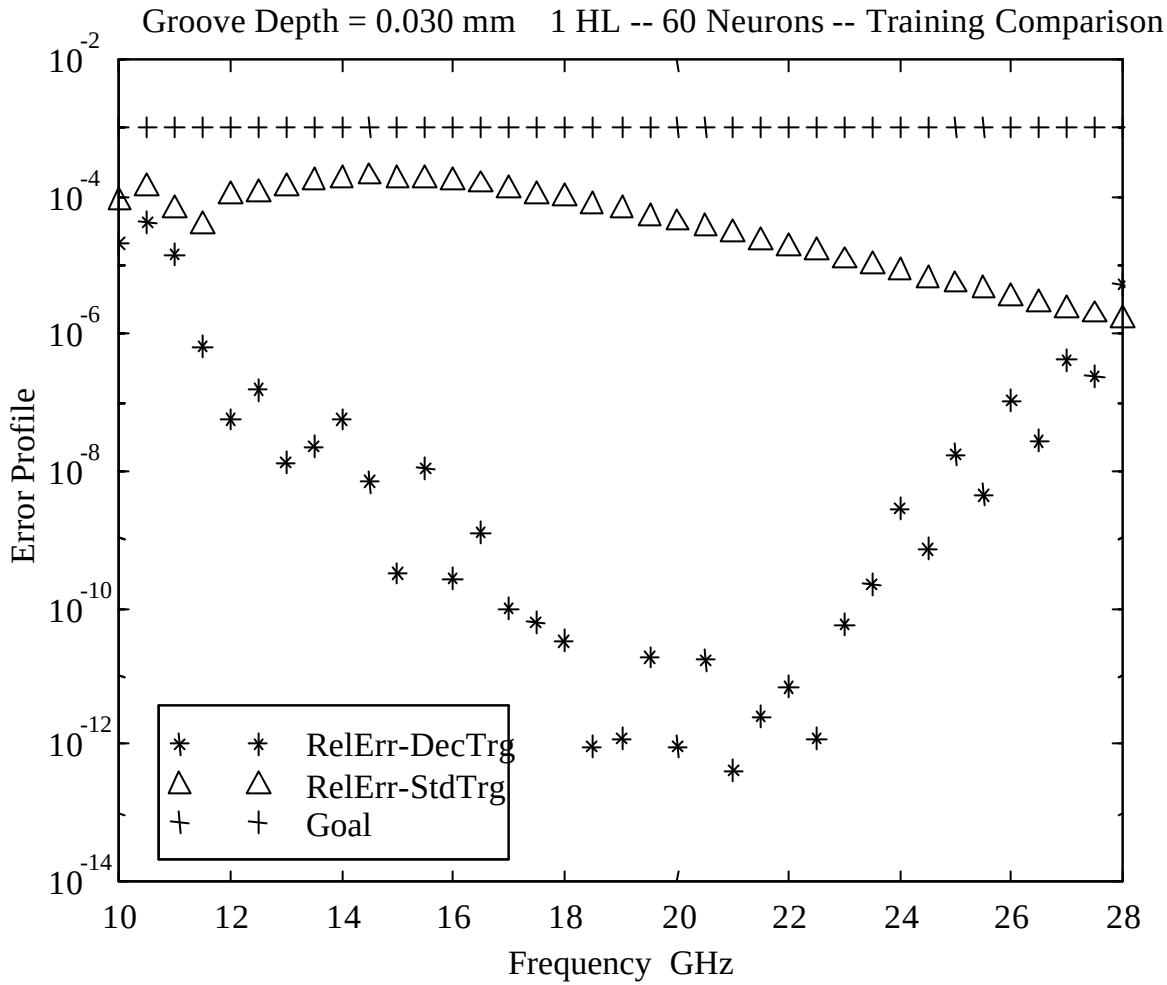
[A. Chubukjan, "Computational Aspects in Modelling Electromagnetic Field Parameters in Microstrips, " Ph.D. Thesis, University of Ottawa, 2000]

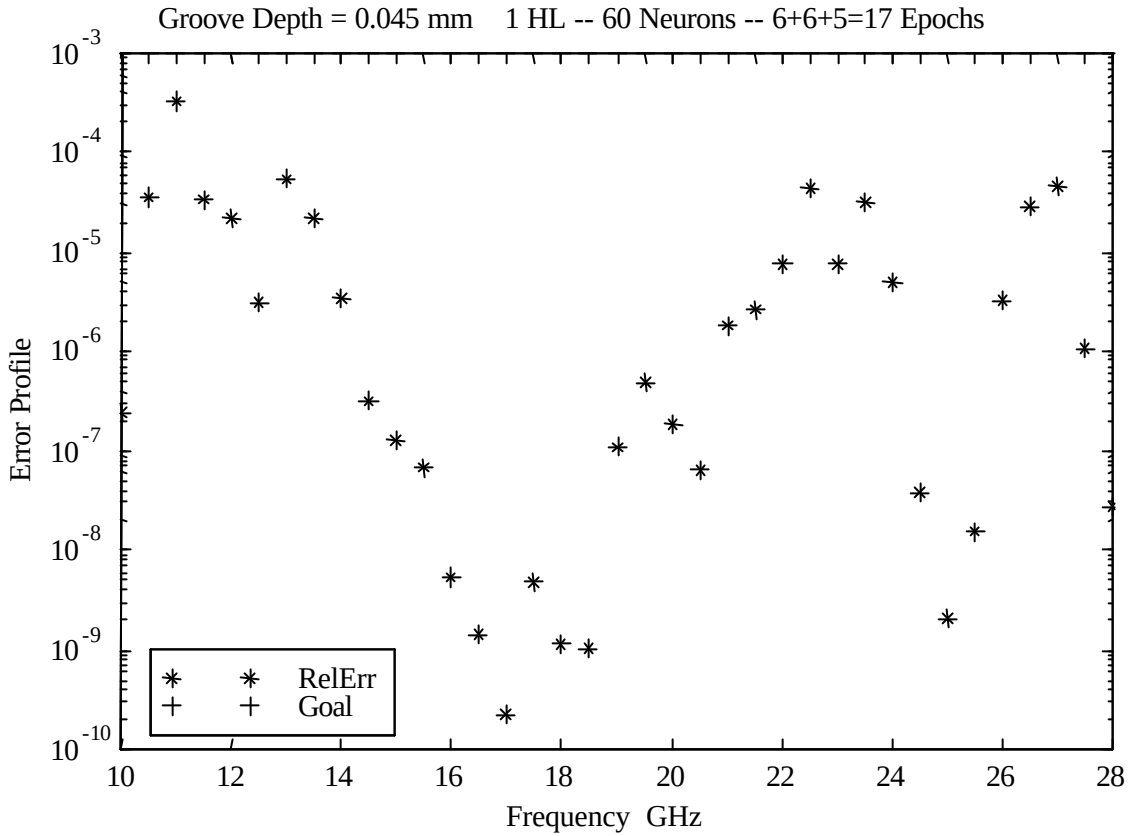
- The problem was solved by "Vector Finite Element Method" VFEM, and the values of the microstrip characteristic impedance for both plain and grooved geometries were obtained. These values describing both the frequency-dependent and/or groove-dependent behaviour of each microstrip geometry were used to train the NN models.

- A feedforward network with backpropagation, having one or more hidden layers with non-linear transfer functions and one output layer with a linear transfer function, is capable of approximating any function with a finite number of discontinuities with arbitrary accuracy. A two-layer sigmoid/linear NN can represent any functional relationship between inputs and outputs if the sigmoid layer has enough neurons.

- The grooved microstrip was modelled initially by two separate one hidden-layer NN architectures, having 50 and 60 hidden neurons respectively. These networks were trained both by decimation and by the standard way. The resulting error obtained by decimation was comparable to that obtained by standard training, and at times, was superior. The networks reached the desired error goal easily, with excellent sum-squared error figures. Nevertheless, the NN architecture with 60 neuron hidden-layer gave better results compared to the 50 neuron hidden-layer architecture, and it was selected for further modelling.

Error performance for standard and decimated training of a “60 neuron one hidden-layer” NN model of grooved microstrip.





Error performance for standard and decimated training of a "60 neuron one hidden-layer" NN model of grooved microstrip.

MODEL CALIBRATION

The whole idea of virtual prototyping relies on the ability to develop ***models conformable to the physical objects and phenomena*** which represent reality very closely.

There is a need for ***calibration techniques able to validate the conformance with the physical reality of the models*** incorporated in the new prototyping tools.

Experimental Measurements

- The EM field training data are conveniently obtained as analytical estimations of far-field values in 3D space and frequency from near-field data using the finite element method combined with method of integral absorbing boundary conditions.
- The near field data could be obtained analytically and/or by physically measuring EM field values at for given frequency values and 3D space locations.
- This approach allows to replace the usual cumbersome open site far-field measurement technique by anechoic chamber measurements.

The amount and extent of the area of measurements is significantly reduced by collecting data in the near-field only and calculating then the far-field values using Poggio's equation:

$$H(r') = \frac{1}{4p}\int_{s1}\left[G(r,r')\frac{\partial H(r)}{\partial n} - H(r)\frac{\partial G(r,r')}{\partial n}\right] dS_1$$

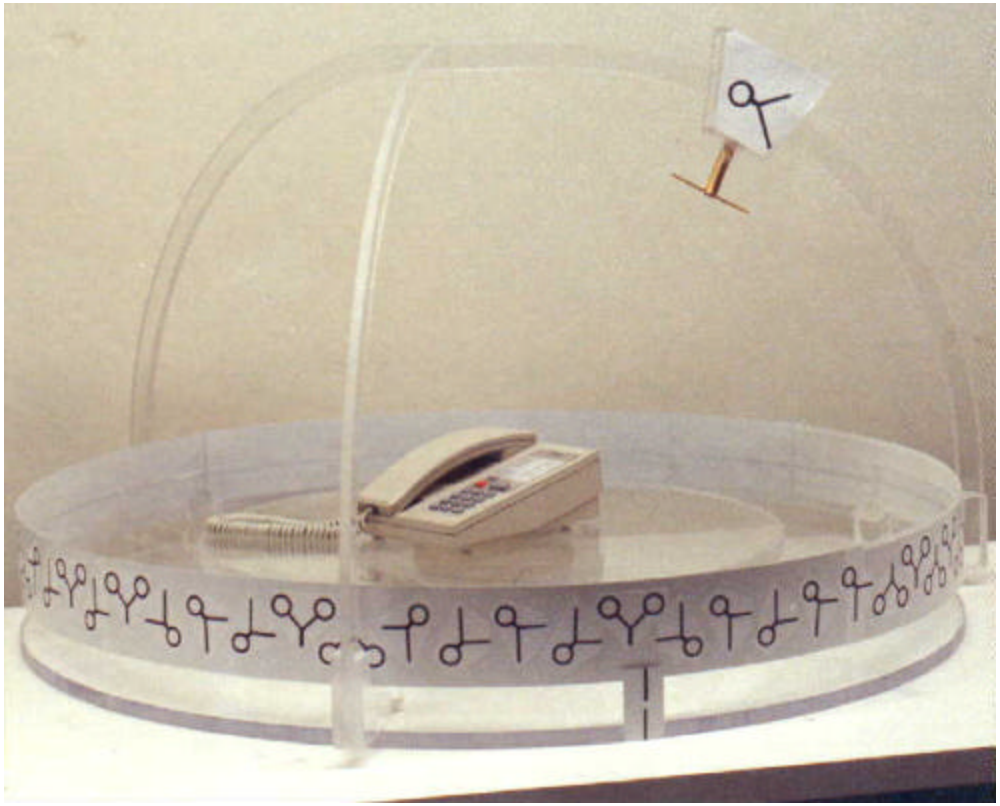
where:

-S_1 is the surface on which measurements are made, closed or made closed,

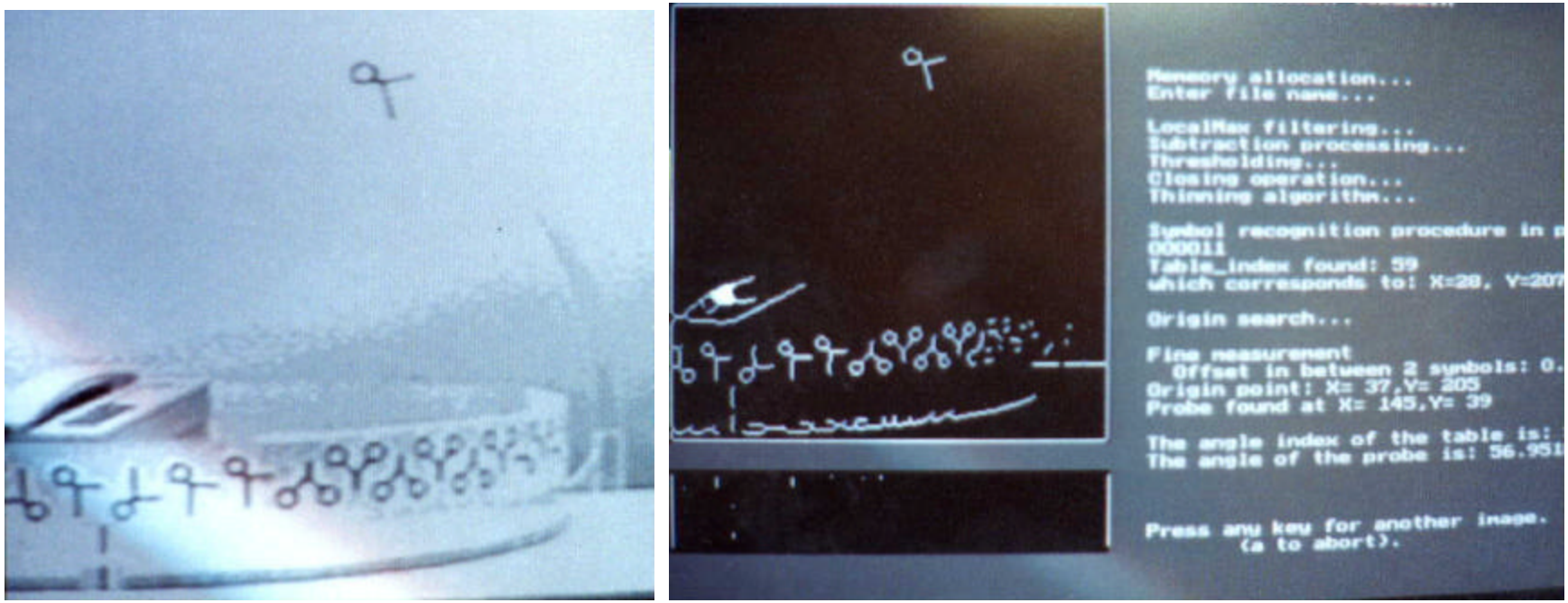
-n is the normal to S_1 and

is the free space Green's function.

- *This equation states that if the field values and their derivatives are known on a closed surface enclosing all inhomogeneities, then the field outside the surface can be calculated.*

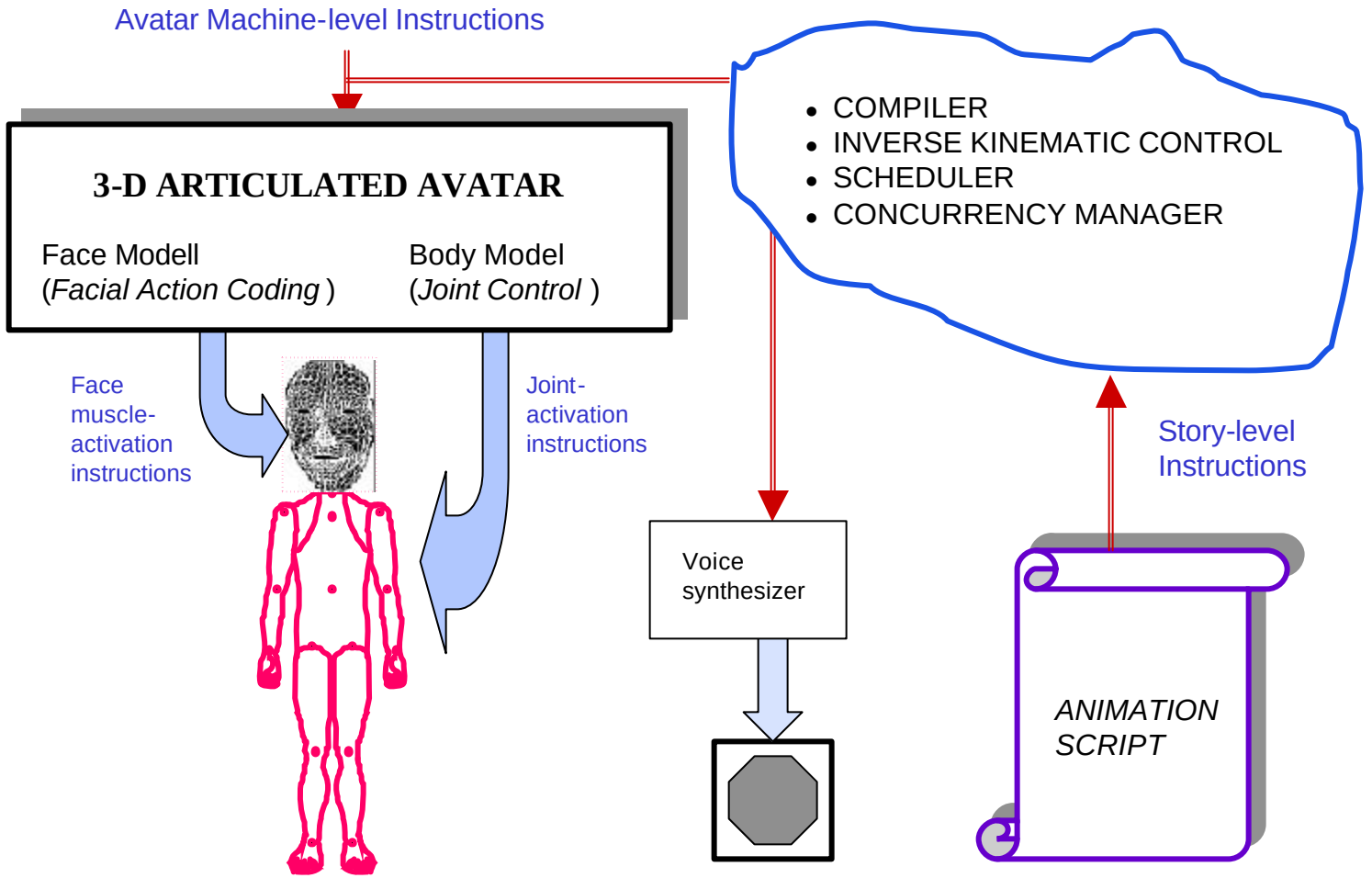


Experimental setup for the noninvasive measurement of the 3D near field data



Computer vision recovery of the 3D position of the EM probe

NEURAL NETWORK MODELLING OF HUMAN AVATARS IN INTERACTIVE VIRTUAL ENVIRONMENTS



Scripting Language: Abstraction Levels

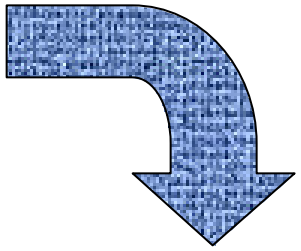
- Three levels of abstraction for the avatar animation scripting language:
  - Highest: story-level description
    - constrained English-like description
    - syntactic and semantic analysis to extract information such as: main player(s), action, subject and object of the action, relative location, degree, etc.
    - translate in a set of skill-level instructions, that may be executed sequentially or concurrently
  - Middle: skill-level macro-instructions
    - describe basic body and facial skills (such as walk, smile, wave hand, etc.)
    - each skill involves a number of muscle/joint activation instructions that may be executed sequentially or concurrently
  - Lowest: muscle/joint activation instructions
    - activation of individual muscles or joints to control the face, body or hand movement

Personalizing Skills

- Add “personality” to skill-level macro-instructions
  - different avatars may perform a certain skill in a “personalized” way
    - examples: “walk like Charlie Chaplin”
      “write like Emil”
  - there is a skill generalization/specialization relationship (similar to object-oriented systems) between
    - a generic skill
    - one or more specialized (or personalized) skills
- Personalizing skills
  - by using Neural Network models
    - off-line training
    - on-line rendering

STORY-LEVEL DESCRIPTION

…..
DanielA sits on the red chair.
DanielA writes "Hello" on stationary.
DanielA sees HappyCat under the white table and starts smiling.
HappyCat grins back.
……



SKILL-LEVEL ("MACRO") INSTRUCTIONS

…..
DanielA's right hand moves the pen to follow the trace representing "H".
DanielA's right hand moves the pen to follow the trace representing "e".
DanielA's right hand moves the pen to follow the trace representing "l".
DanielA's right hand moves the pen to follow the trace representing "l".
DanielA's right hand moves the pen to follow the trace representing "o".
……

SKILL-LEVEL MACRO-INSTRUCTIONS

…

DanielA's right hand moves the pen to follow the trace representing "H".

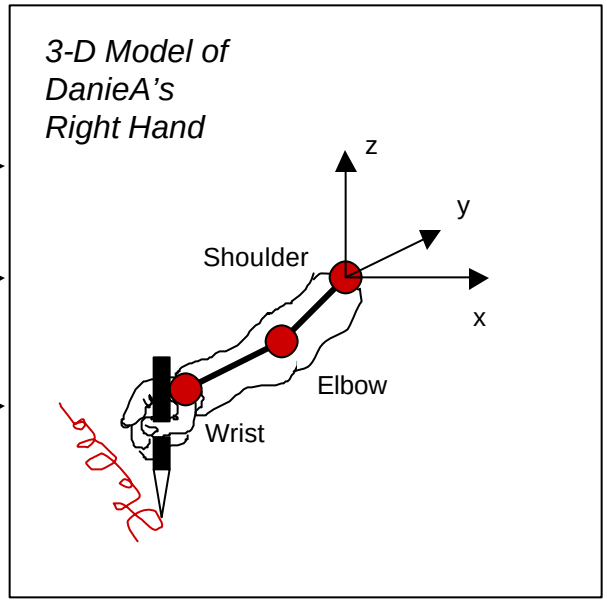
…

DanielA's specific style of moving his right arm joints to write "H" (**NN model capturing DanielA's writing personality**)

Rotate Wrist to α^{i}

Rotate Elbow to β^{j}

Rotate Shoulder to γ^{k}



References

- W. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133, 1943.
- D.O. Hebb, *The Organization Of Behavior*, Wiley, N.Y., 1949.
- J. von Neuman, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in *Automata Studies*, (C.E. Shannon, Ed.), Princeton, NJ, Princeton University Press,1956.
- F. Rosenblat, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, Vol. 65, pp. 386-408, 1958.
- B. Widrow and M.E. Hoff, "Adaptive Switching Circuits," *1960 IRE WESCON Convention Record, IRE Part 4*, pp. 94-104, 1960.
- M. Minski and S. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1969.
- J.S. Albus, "A Theory of Cerebellar Function," *Mathematical Biosciences*, Vol. 10, pp. 25-61, 1971.
- T. Kohonen, "Correlation Matrix Memories," *IEEE Tr. Comp*., Vol. 21, pp. 353-359, 1972.
- J. A. Anderson, "A Simple Neural Network Generating an Interactive Memory," *Mathematical Biosciences*, Vol. 14, pp. 197-220, 1972.
- S. Grossberg, "Adaptive Pattern Classification and Universal Recording: I. Parallel Development and Coding of Neural Feature Detectors," *Biological Cybernetics*, Vol. 23, pp.121-134, 1976.
- J.J. More, "The Levenberg-Marquardt Algorithm: Implementation and Theory," in Numerical Analysis, pp. 105-116, Spriger Verlag, 1977.
- K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Tr. Syst. Man Cyber*., Vol. 13, No. 5, pp. 826-834, 1983.

- D. E. Rumelhart, G.E. Hinton, and R.J. Willimas, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, (D.E. Rumelhart and J.L. McClelland, Eds.,) Vol.1, Ch. 8, MIT Press, 1986.
- D.W. Tankand J.J. Hopefield, "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," *IEEE Tr. Circuits Systems*, Vol. 33, No. 5, pp. 533-541, 1986,
- M.J.D. Powell, "Radial Basis Functions for Multivariable Interpolation" A Review," in *Algorithms for the Approximation of Functions and Data*, (J.C. Mason and M.G. Cox, Eds.), Clarendon Press, Oxford, UK, 1987.
- G.A. Carpenter and S. Grossberg, "ART2: Self-Organizing of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, Vol. 26, No. 23, pp. 4919-4930, 1987.
- B. Kosko, "Bidirectional Associative Memories," *IEEE Tr. Syst. Man Cyber.*, Vol. 18, No. 1, pp. 49-60, 1988.
- T. Kohonen, *Self_Organization and Associative Memory*, Springer-Verlag, 1989.
- K. Hornic, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks*, Vol. 2, pp. 359-366, 1989.
- B. Widrow and M.A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proc. IEEE*, pp. 1415-1442, Sept. 1990.
- B. Kosko, *Neural Networks And Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall, 1992.
- E. Sanchez–Sinencio and C. Lau, (Eds.), *Artificial Neural Networks*, IEEE Press, 1992.
- A. Hamilton, A.F. Murray, D.J. Baxter, S. Churcher, H.M. Reekie, and L. Tarasenko, "Integrated Pulse Stream Neural Networks: Results, Issues, and Pointers," *IEEE Trans. Neural Networks*, vol. 3, no. 3, pp. 385-393, May 1992.
- S. Haykin, *Neural Networks: A Comprehensive Foundation*, MacMillan, New York, 1994.
- M. Brown and C. Harris, *Neurofuzzy Adaptive Modelling and Control*, Prentice Hall, NY, 1994.

- C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, NY, 1995
- M.T. Hagan, H.B. Demuth, and M. Beale, *Neural Network Design*, PWS Publishing Co., 1995
- S. V. Kartalopoulos, *Understanding Neural and Fuzzy Logic:Basic Concepts and Applications*, IEEE Press, 1996.
- M. T. Hagan, H.B. Demuth, M. Beale, *Neural Network Design*, PWS Publishing Co., 1996.
- C.H. Chen (Editor), *Fuzzy Logic and Neural Network Handbook*, McGraw Hill, Inc., 1996.
- ***, "Special Issue on Artificial Neural Network Applications," *Proc. IEEE*, (E. Gelenbe and J. Barhen, Eds.), Vol. 84, No. 10, Oct. 1996.
- J.-S.R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, 1997.
- C. Alippi and V. Piuri, Neural Methodology for Prediction and Identification of Non-linear Dynamic Systems, " in *Instrumentation and Measurement Technology and Applications*, (E.M. Petriu, Ed.), pp. 477-485, IEEE Technology Update Series, 1998.
- ***, "Special Issue on Pulse Coupled Neural Networks," *IEEE Tr. Neural Networks*, (J.L. Johnson, M.L. Padgett, and O. Omidvar, Eds.), Vol. 10, No. 3, May 1999.
- C. Citterio, A. Pelagotti, V. Piuri, and L. Roca, "Function Approximation – A Fast-Convergence Neural Approach Based on Spectral Analysis, *IEEE Tr. Neural Networks*, Vol. 10, No. 4, pp. 725-740, July 1999.
- ***, "Special Issue on Computational Intelligence," *Proc. IEEE*, (D.B. Fogel, T. Fukuda, and L. Guan, Eds.), Vol. 87, No. 9, Sept. 1999.
- L.I. Perlovsky, *Neural Networks and Intellect, Using Model-Based Concepts*, Oxford University Press, NY, 2001.