Appl. Math. and Comp. Sci., 1995, vol.5, No.1, 105–138

ANALOG NEURAL NETWORKS FOR SOLVING IN REAL-TIME LINEAR INVERSE AND TOTAL LEAST SQUARES PROBLEMS

ANDRZEJ CICHOCKI*, TADEUSZ KACZOREK**
JANUSZ MAZUREK***

A class of simplified low-cost artificial neural networks with on-line adaptive learning algorithms are discussed for solving large system of algebraic equations and related problems in real-time. The proposed learning algorithms for Least Squares (LS), Total Least Squares (TLS) and Data Least Squares (DLS) problems can be considered as an extension and generalization of the well known Least Mean Squares (LMS) and Kaczmarz algorithms. A generalized maximum entropy and minimum p-norm criteria are used as a principle for reconstructing images and/or signals from noisy and incomplete projection data. The inverse problem is reformulated as a suitable optimization problem and solved by a unified neural network.

1. Introduction

Machine intelligence will be probably the dominant technology in the near future. Artificial Neural Networks (ANN) are important components of intelligent systems. ANN's are today undoubtely one of the fastest growing areas of research. ANN's are not composed of one approach but are rather a broad body of often loosy related knowlegde, tools and techniques. The main objective of this paper is the development of adaptive algorithms for solving a wide class of inverse problems and total least squares problems.

Many problems in science and technology involve solving a large system of linear equations (Cichocki and Kaczorek, 1992a; 1992b; Cichocki and Unbehauen, 1992; 1994a; 1994b; Cichocki *et al.*, 1992; 1995; Kaczmarz, 1937; Osowski, 1993; Lillo *et al.*, 1993)

$$Ax \cong b \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$ is a data matrix, $b \in \mathbb{R}^m$ is an observation vector. Usually A and b are perturbated versions of the exact but unobservable matrices A_0, b_0, respectively, i.e. the exact relation is described by the matrix equation

$$A_0 x = b_0 \tag{2}$$

* Institute of the Theory of Electrical Engineering and Measurement, Warsaw University of Technology, Poland, e-mail: cia@iem.pw.edu.pl
** Institute of Control and Industrial Electronics, Warsaw University of Technology, ul. Koszykowa 75, 00–662 Warsaw, Poland
*** Technical University of Wrocław, Poland; Neurolab, Germany

Such a description appears in a broad class of scientific and engineering problems. For example, in image and signal restoration the matrix A is the degradation matrix, in deconvolution problems it is the convolution matrix; in wave propagation problems, in electromagnetics and acoustics, it is the propagation matrix. Moreover, every linear parameter estimation problem arising in automatic control, system identification, signal processing, physics, biology and medicine gives rise to an overdetermined set of linear equations (1) (Cichocki and Unbehauen, 1994a; DeGroat and Dowling, 1993; Marrian and Peckerar, 1989; Osowski, 1993). The system of equations is sometimes underdetermined due to the lack of information, but often it is greatly overdetermined and usually inconsistent (i.e. it is self-contradictory) due to errors, noise and other perturbations. Generally speaking, in signal processing applications the overdetermined case ($m > n$) describes filtering, estimation of parameters, enhancement, deconvolution and identification problems, while the underdetermined ($m < n$) case describes inverse and extrapolation problems (Cichocki and Unbehauen, 1994a).

In many applications (e.g. robotics, signal processing, automatic control) an online (i.e. in real time) solution of a system of linear equations is desired. For such real time applications, when the solution is to be obtained within a time of the order of hundred nanoseconds a digital computer often cannot comply with the desired computation time or its use is too expensive. One possible approach for solving such a problem is to employ analog artificial neural networks (Cichocki and Unbehauen, 1994a). However, known methods and network architectures require to use at least n (n is the number of unknown variables) artificial neurons (processing units) (Cichocki and Unbehauen, 1992; 1994a; Hopfield 1984; Kennedy and Chua, 1988; Marrian and Peckerar, 1989; Tank and Hopfield, 1986). In many engineering applications, e.g. in image reconstruction and in computer tomopraphy it is required to solve very large systems of linear algebraic equations with high speed and throughput rate (Herman, 1980; Herman *et al.*, 1991; Kak and Slaney, 1987; Lu *et al.*, 1992; Madych, 1991; Rosenfield and Kak, 1981; Wang and Lu, 1992) For such problems, a known neural network architecture requires an extremely large number of processing units (artificial neurons) so that a practical hardware implementation of the neural network may be difficult, expensive and even impossible.

In this paper we will propose a new analog artificial neural network containing only one simplified neuron (single adaptive processing unit) with an on chip implemented adaptive learning algorithm. Many researchers in different scientific desciplines have developed a class of efficient adaptive algorithms for solving systems of linear equations and related problems. As early as 1937 a Polish mathematician (Banach School) Kaczmarz (1937) developed a very simple and efficient Row-Action Projection (RAP) algorithm. Since the systems of linear equations have been used in many diverse disciplines and problems, the Kaczmarz RAP algorithm and its modifications have been independently and repeatedly rediscovered and in fact they appear under different names in different applications. For example, in the field of medical imaging for computerized tomography it is often called an Algebraic Reconstructed Technique (ART) (Censor *et al.*, 1989); in the adapative filtering literature it is known as the α-LMS algorithm or Widrow-Hoff delta rule (Cichocki and Unbehauen, 1994a).

The main objective of this paper is analog circuit design of a neural network for implementing such adaptive algorithms. The second objective is to propose some extensions and modifications of the existing adaptive algorithms. The third objective is to demonstrate the validity and high performance of the proposed neural network models by computer simulation experiments.

2. Formulation of the Problems

In general, without reference to any specific application, the overdetermined problem can be stated as follows. Let us assume that we want to solve a set of linear algebraic equations written in scalar form as

$$\sum_{j=1}^{n} a_{ij}x_j \cong b_i \quad (i = 1, 2, \ldots, m) \text{ with } m \geq n \text{ typically } m \gg n \tag{3a}$$

or in the matrix form

$$Ax \cong b \tag{3b}$$

Here, x is the n-dimensional unknown vector, b is the m-dimensional observation vector and $A = [a_{ij}]$ is the $m \times n$ real coefficient matrix with known elements. Note that the number of equations is generally not restricted to n; it can be less than, equal to or greater than the number of variables. In the ordinary Least Squares (LS) approach to problem (3a)–(3b) the measurements in the matrix A are assumed to be free from error and all errors are confined to the observation vector b. In this approach one defines a cost (error function) $E(x)$ as

$$\begin{aligned} E(x) &= \frac{1}{2}\|Ax - b\|_2^2 + \frac{1}{2}\nu\|x\|_2^2 = \frac{1}{2}(Ax - b)^T(Ax - b) + \frac{1}{2}\nu\, x^T x \\ &= \frac{1}{2}\sum_{i=1}^{m} r_i^2(x) + \frac{1}{2}\nu\sum_{j=1}^{n} x_j^2 \end{aligned} \tag{4}$$

where the upper index T denotes the transposition, the residuals are given as

$$r_i(x) = a_i^T x - b_i = \sum_{j=1}^{n} a_{ij}x_j - b_i$$

and $\nu \geq 0$ is the regularization parameter. The first term is the standard least squares term and it forces the sum of square residuals to be minimal. The second term is the regularization term whose purpose is to force a smoothness constraint on the estimated solution x^* for ill-conditioned problems. The regularization parameter determines the relative importance of this term (Cichocki and Unbehauen, 1994a). Using a standard gradient approach for the minimization of the cost function the problem can be mapped to the system of linear differential equations

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -\mu\left[A^T(Ax - b) + \nu x\right] \tag{5a}$$

with any initial conditions $x(0) = x^{(0)}$ (typically $x(0) = 0$), where

$$\mu = \text{diag}(\mu_1, \mu_2, \ldots, \mu_n), \qquad \mu_j > 0 \quad \forall\, j$$

Note that the direct implementation of system (5a) requires the use of $m + n$ linear processing units (Cichocki and Unbehauen, 1994a). In fact, the number of processing units can be reduced to n units since eqn. (5a) can be simplified as

$$\frac{dx}{dt} = -\mu\Big[(Wx - \theta) + \nu x\Big] \tag{5b}$$

where

$$W = A^T A, \qquad \theta = A^T b$$

but this requires extra precalculations and it is inconvenient for large matrices especially when the entries a_{ij} and/or b_i are slowly changing in time (i.e. are time variable). In the next section we will show how we can avoid this disadvantage. In other words, the known neural network realizations for solving a linear Least Squares (LS) problem requires (cf. eqns. (5a)–(5b)) an excessive number of building blocks (analog multipliers and summers). In the next section we will propose a new approach which enables us to solve the LS problem more efficiently and econonomically.

The ordinary LS problem is optimal only if all errors are confined to the observation vector b and they have the Gaussian distribution. The measurements in the data matrix are assumed to be free from errors. However, such an assumption is often unrealistic (e.g. in image recognition and computer vision) since sampling errors, modeling errors and instrument errors may imply noise inaccuracies of the data matrix A (Cichocki and Unbehauen, 1994b). The Total Least Squares problem (TLS) has been devised as a more global and often more reliable fitting method than the standard LS problem for solving an overdetermined set of linear equations when the measurements in b as well as in A are subject to errors.

Especially, if the errors are (approximately) uncorrelated with zero a mean and equal variance, the TLS solution is more accurate than the LS solution (Golub and Van Loan, 1980; 1989; Van Huffel and Vandewalle, 1989; 1991). Golub and Van Loan (1989) were the first to introduce the TLS approach into the field of numerical analysis and they developed an algorithm based on the Singular Value Decomposition (SVD) (Golub and Van Loan, 1980; 1989). Van Huffel and Vandewalle (1989; 1991) investigated the problem, presented a detailed analysis and extended the analysis to cover the non-generic TLS case.

While the LS problem confines all errors to the observation vector b, the TLS problem assumes a perturbation of the data matrix A (by a matrix ΔA) and the observation vector b (by a vector) and tries to compensate for these. The TLS problem can be formulated as the optimization problem to find the vector x^*_{TLS} that minimizes

$$\|\Delta A\|_F^2 + \|\Delta b\|_F^2 \tag{6}$$

subject to the equality constraint

$$(A - \Delta A)x^*_{\text{TLS}} = (b - \Delta b)$$

where $\|\Delta\|_F$ denotes the Frobenius norm of Δ. In other words, the TLS problem seeks to

$$\begin{aligned} &\underset{[A_0;b_0]\in\mathbb{R}^{m\times(n+1)}}{\text{minimize}} \ \|[A;B]-[A_0;b_0]\|_F \\ &\text{subject to range } (b_0)\subset \text{range } (A_0) \end{aligned} \tag{7}$$

The main tool for solving the TLS problem is the singular value decomposition (SVD) of the matrix A and the extended matrix $[A, b]$. Let

$$v_{n+1} = [v_{1,n+1}, v_{2,n+1}, \ldots, v_{n+1,n+1}]^T$$

be the right singular vector corresponding to the smallest singular value σ_{n+1} of the extended matrix $[A, b]$. The TLS solution x^*_{TLS} is obtained as (Golub and Van Loan, 1980).

$$x^*_{\text{TLS}} = -\frac{1}{v_{n+1,n+1}}[v_{1,n+1}, v_{2,n+1}, \ldots, v_{n+1,n+1}]^T \tag{8}$$

As the singular value σ_{n+1} goes to zero, the LS and TLS solutions approach each other.

It is important to point out that the ordinary LS problem minimizes the sum of the squared residuals

$$r_i(x) = \sum_{j=1}^{n} a_{ij}x_j - b_i \tag{9}$$

while the TLS problem minimizes a sum of the weighted residuals

$$r_{\text{TLS}}(x) = \frac{(Ax-b)}{(1+x^Tx)^{\frac{1}{2}}} \tag{10}$$

In other words, the TLS problem can be equivalently formulated as the minimization (with respect to the vector x) of the cost function

$$E_{\text{TLS}}(x) = \sum_{i=1}^{m}\left(\frac{r_i(x)}{\sqrt{1+x^Tx}}\right)^2 \tag{11}$$

In comparison with the standard LS approach, it is generally quite burdensome and very time consuming to obtain the solution of the TLS problem (Van Huffel and Vandewalle, 1991). This is probably why the TLS approach has not been as widely used as the usual LS approach although the TLS approach was investigated in robust statistics long ago. We will propose a very simple and efficient algorithm (or more precisely a class of algorithms) to solve the TLS problem by using only one single artificial neuron.

The underdetermined problem usually has an infinite number of solutions. In order to find optimal unique solution the problem is formulated as the following optimization problem

$$\text{minimize } f(x) \tag{12a}$$

subject to the equality constraints

$$Ax = b \tag{12b}$$

and eventually to the inequality box (simple bounds) constraints

$$x_{j\,\min} \leq x_j \leq x_{j\,\max} \tag{12c}$$

(typically $x_j \geq 0$), where $f(x)$ is a suitable cost (objective) function.

Sometimes the optimization problem is formulated as

$$\text{minimize } f(x) \tag{13a}$$

subject to the equality constraints

$$Ax \leq b \tag{13b}$$

and

$$x_{j\,\min} \leq x_j \leq x_{j\,\max} \quad \forall\, j \qquad (j = 1, 2, \ldots, n) \tag{13c}$$

The choice of equality constraints (12b) or inequality constraints (13b) depends on the character of the noise (Censor *et al.*, 1989; Frieden and Zoltani, 1985; Herman 1980; Herman *et al.*, 1991; Kak and Slaney, 1987; Lu *et al.*, 1992; Madych, 1991; Rosenfield and Kak, 1981; Wang and Lu, 1992).

In practice, two classes of criteria or two classes of objective functions $f(x)$ are often used:

1) The minimum p-norm criteria,

2) The maximum entropy criteria.

For the minimum p-norm problem the objective function can take the form:

$$f(x) = \frac{1}{p} \sum_{j=1}^{n} |x_j|^p \tag{14}$$

for $1 \leq p < \infty$.

We are particularly interested in the following three special cases:

a) the minimum 1-norm problem, sometimes called minimum fuel problem,

b) the standard minimum 2-norm problem, sometimes called minimum energy problem,

c) the minimum infinity (Chebyshev)-norm problem, called minimum amplitude problem which minimizes:

$$f(x) = \max_j \left\{ |x_j| \right\} = \|x\|_\infty \tag{15}$$

subject to $Ax = b$.

It is interesting to note that for the standard minimum 2-norm problem, i.e. for

$$f(x) = \frac{1}{2}\|x\|_2^2 = \frac{1}{2}\sum_{j=1}^{n} x_j^2 \tag{16}$$

subject to $Ax = b$, there are algorithms based on linear algebra which may solve the problem explicitly (Cichocki and Unbehauen, 1994a).

It is well known that the optimal solution x^* can be explicitly expressed as

$$x^* = A^T(AA^T)^{-1}b = A^+b \tag{17a}$$

or

$$x^* = \lim_{\nu \to 0}(\nu I + AA^T)^{-1}A^Tb = (AAT)^+A^Tb = A^+b \tag{17b}$$

(where A^+ means the pseudo-inverse matrix of A) and the resulting value of the 2-norm reads

$$\|x^*\|_2 = \left(b^T(AA^T)^{-1}b\right)^{1/2} \tag{18}$$

However, for image reconstruction problems the matrices A are typically too large to allow us to use the above formulas[1]. Moreover, for $p \neq 2$ the minimum p-norm problems are somewhat more complex and we look rather for one unifying and generalized algorithm to solve problem (12) or (13).

For maximum entropy estimation problems the objective function is defined as (Censor *et al.*, 1989)

$$f(x) = -S(x) \tag{19}$$

where $S(x)$ is a suitable entropy measure. Here, the entropy function $S(x)$ is taken with the minus sign in order to have a minimization problem instead of the maximization problem.

Three common choices of the entropy are (Censor *et al.*, 1989; Frieden and Zoltani, 1985):

1) Shannon's entropy

$$S_H(x) = -\sum_{j=1}^{n} x_j \ln x_j \quad \text{with} \quad x_j > 0 \tag{20a}$$

2) Burg's entropy

$$S_B(x) = -\sum_{j=1}^{n} \ln x_j \qquad \text{with} \quad x_j > 0 \tag{20b}$$

3) Friden's and Zoltani's entropy

$$S_{FZ}(x) = -\sum_{j=1}^{n} \left[x_j \ln x_j + (x_{j\max} - x_j)\ln(x_{j\max} - x_j)\right] \tag{20c}$$

with constraints $0 < x_j < x_{j\max} \ \forall\, j$.

[1] Typically m and n are in the order of hundreds or even thousands (see Example 4)

The maximum entropy estimation criteria are attractive alternatives to the minimum p-norm estimation measure with a solid theoretical background (Censor *et al.*, 1989; Frieden and Zoltani, 1985; Herman 1980; Herman *et al.*, 1991; Kak and Slaney, 1987; Lu *et al.*, 1992; Madych, 1991; Rosenfield and Kak, 1981; Wang and Lu, 1992). The main idea is to estimate the variables (signals) which satisfy specified constraints and to provide a maximum entropy. Roughly speaking, the maximum entropy criteria enable us to find estimates of a vector x that provides minimum information or in the absence of *a prior* knowledge assumes as small value as possible.

An extensive literature is available, which theoretically and also experimentally justifies the use of objective functions $f(x)$ (cf. eqns. (14), (15), (20a)–(20c)). In image reconstruction, the application of these criteria has been proposed on a more experimental basis. It is out of the scope of this paper to compare the various criteria and associated objective functions. They all have some theoretical basis and are related to the maximum likelihood estimation. Our main purpose is to develop new "general-purpose" neural network architectures and associated algorithms for solving optimization problems (11), (12) or (13). Since optimization problems (12) and (13) are of quite a general nature and arise in many scientific problems, e.g. in optical image resoration, signal reconstruction, high resolution spectrum estimation, beam forming, extrapolation, control of discrete systems etc., we will carry our futher considerations without any specific applications.

3. Neural Networks Models for Underdetermined Problems

3.1. General Unified Model-Standard Approach

The mapping of a constrained optimization problem into an appropriate energy (cost) function is the standard (commonly) applied strategy in the design of neural networks (Cichocki and Unbehauen, 1994a; Hopfield, 1984; Kennedy and Chua, 1988; Marrian and Peckerar, 1989; Tank and Hopfield, 1986). In other words, in order to formulate optimization problem (12) or (13) in terms of artificial neural network, the key step is to construct an appropriate energy (cost) function $E(x)$ so that the lowest energy state will correspond to the desired estimate (optimal solution) x^*. The construction of a suitable energy function enables us to transform the minimization problem into a system of differential or difference equations on the basis of which we can design an associated artificial neural network with appropriate connection weights (synaptic strength) and input excitations.

For optimization problems (12) and (13) we can construct the general energy function (on the basis of the penatly method) (Cichocki and Unbehauen, 1994a)

$$E(x) = \nu f(x) + \sum_{i=1}^{m} P\Big(r_i(x)\Big) \tag{21}$$

with $x_{j\,\min} \leq x_j \leq x_{j\,\max} \;\; \forall\, j$, where $\nu > 0$,

is the scaling coefficient called penalty parameter. $P(r_i)$ are the penalty function terms and $r_i(x)$ are the residuals defined as

$$r_i(x) = a_i^T x - b_i = \sum_{j=1}^{n} a_{ij} x_j - b_i, \qquad i = 1, 2, \ldots, m \tag{22}$$

Exemplary penalty function terms for equality constraints (2b) can take, e.g. one of the following forms (see Fig. 2):

$$P(r) = \frac{1}{2}r^2 \quad \text{(quadratic)} \tag{23a}$$

$$P(r) = \frac{1}{\rho}|r|^{\rho}, \quad \rho \geq 1 \quad (\rho \text{ is the order of growth}) \tag{23b}$$

$$P(r) = \begin{cases} \dfrac{r^2}{2} & \text{for } |r| \leq \beta \\ \beta|r| - \dfrac{\beta^2}{2} & \text{for } |r| > \beta \end{cases} \tag{23c}$$

$$P(r) = \beta^2 \ln \cosh\left(\frac{r}{\beta}\right), \quad \beta > 0 \tag{23d}$$

$$P(r) = \frac{1}{\rho}|r|^{\rho} + \frac{1}{2}r^2, \quad \rho > 1 \quad \text{e.g. } \rho = \frac{9}{8} \tag{23e}$$

For inequality constraints (13b) the penalty function terms are typically defined as

$$P(r) = \begin{cases} \dfrac{1}{2}r^2 & \text{if } r > 0 \\ 0 & \text{otherwise} \end{cases} \tag{24a}$$

or

$$P(r) = \begin{cases} \dfrac{K_1}{2}r^2 + \dfrac{K_2}{\rho}|r|^{\rho} & \text{if } r > 0 \\ 0 & \text{otherwise} \end{cases} \tag{24b}$$

with $\rho \geq 1$, $K_1 \geq 0$, $K_2 \geq 0$ (typically $K_1 = K_2 = 1$).

It should be emphasized here that instead of the penalty technique we can employ the Lagrange multiplier or augmented Lagrange multiplier method (Cichocki and Unbehauen, 1994a). However, in order to streamline and simplify our further considerations we limit here our discussion to the energy function given by eqn. (21).

Using the standard gradient descent approach (Cichocki and Unbehauen, 1994a; Tank and Hopfield, 1986) for the minimization of the energy function $E(x)$ the problem can be mapped to a non-linear system of ordinary diferential equations, i.e.

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = -\mu\frac{\partial E(x)}{\partial x_j} \qquad \text{with } \mu > 0 \tag{25}$$

Hence taking into account eqn. (21) we have

$$\frac{dx_j}{dt} = -\mu \left[\nu \varphi(x_j) + \sum_{i=1}^{m} a_{ij} \Psi(r_i) \right] \tag{26}$$

where $\mu_j > 0$ is the learning rate, $\nu > 0$,

$$\varphi(x_j) := \frac{\partial f(x)}{\partial x_j} \quad \text{are the activation functions of the output neurons,}$$

$$\Psi(r_i) := \frac{\partial P(r_i)}{\partial r_i} \quad \text{are the activation functions of the input neurons,}$$

$$i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, n; \quad r_i(x) = a_i^T x - b_i = \sum_{j=1}^{n} a_{ij} x_j - b_i$$

On the basis of the system of differential equations (26) we can design the associated artificial neural network (ANN) with suitable connection weights (synaptic strenght), activation functions and input excitations. The functional block diagram of the ANN is depicted in Fig. 1. The network can be considered as a modification of the well-known Tank-Hopfield model (Hopfield 1984; Kennedy and Chua, 1988; Tank and Hopfield, 1986). The network consists of integrators, adders (summing amplifiers) with associated connection weights and non-linear building blocks realizing the specified activation functions $\varphi(x_j)$ and $\Psi(r_i)$. The connection weights a_{ij} can be fixed or time-variable depending on the entries of the matrix A. In practice, they can be realized, e.g. by using VLSI analog multipliers, tunable (voltage controlled) transconductors, programmable switched-capacitors or by high-resistivity polysilicon thin-film resistors (Cichocki and Unbehauen, 1992; Unbehauen and Cichocki, 1989). However, it is beyond the scope of this paper to discuss Very Large Scale Integration (VLSI) implementations of the proposed neural networks, architectures, since the advance in this field is rapid and many choices are available. We rather concentrate here on developing universal and flexible network architectures so that the user can apply them to his specific applications in practical environments. The network of Fig. 1 consists of two layers of processing units. The first layer called input layer or the "sensor layer" (since it senses the actual variables x_j) computes the actual residuals $r_i(x)$ and actual errors (activation functions) $\Psi(r(x))$. The desired variables x_j are computed in the second layer called the output layer, where the signals $\Psi(r_i)$ are combined integrated in time by analog integrators (Cichocki and Unbehauen, 1994a).

The activation functions $\Psi(r_i)$ and $\varphi(x_j)$ can take different forms. Exemplary plots of the activation functions are shown in Fig. 2a–2d and Fig. 3. It should be noted that simple box (bound) constraints $x_{j\,\min} \leq x_j \leq x_{j\,\max}$ may be fulfilled by employing limiting integrators with non-linear (hardware) limiters as their outputs. This means that the input signal of an integrator is integrated but cannot drive the output x_j beyond the specified limits (cf. Fig. 1). In such an approach all box constraints are "hard", i.e. the constraints must never be violated; neither at the final solution nor during the optimization process. An alternative approach is to employ "soft" box constraints which may be violated somehow during the optimization process (Cichocki and Unbehauen, 1994a).

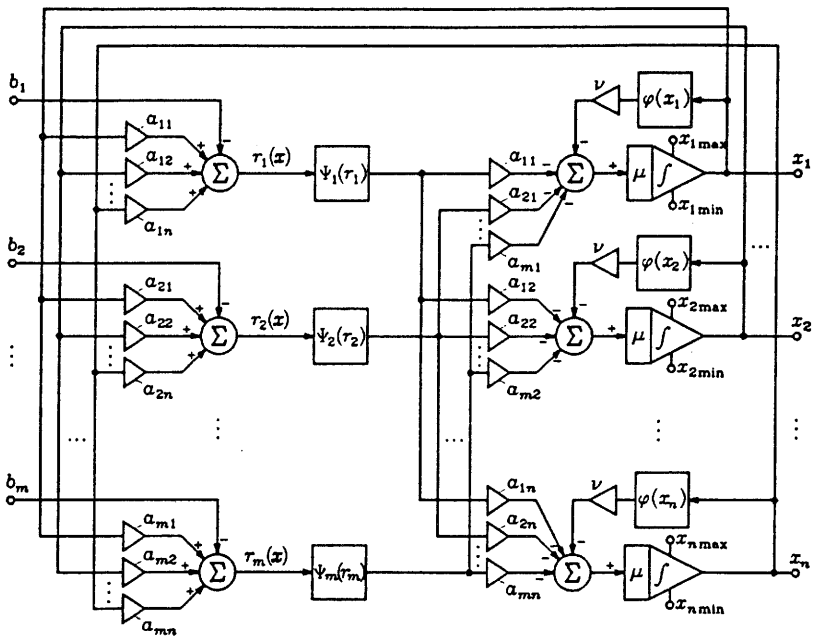


Fig. 1. A neural network architecture for solving the optimization problem (12a)–(12c).

It is interesting to note that the general architecture shown in Fig. 1 can be somewhat simplified for some special cases. Let us consider optimization problem (12) for which we can construct a specific energy function with a quadratic penatly term:

$$E(x) = \nu f(x) + \frac{1}{2}\|Ax - b\|_2^2 = \nu f(x) + \frac{1}{2}\sum_{i=1}^{m} r_i^2(x) \tag{27}$$

with $\nu > 0$, $r_i(x) = a_i^T x - b_i$.

Minimizing energy function (17) leads to the system of differential equations

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = -\mu\left[\nu\varphi(x_j) + \sum_{i=1}^{m} a_{ij} r_i(x)\right] \tag{28}$$

where $\mu > 0$.

Taking into account that

$$r_i(x) = a_i^T x - b_i = \sum_{j=1}^{n} a_{ij} x_j - b_i \tag{29}$$

the system of differential equations can be written as

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = -\mu\left[\nu\varphi(x_j) + \sum_{k=1}^{n} w_{kj} x_k - \theta_j\right] \tag{30a}$$

where $w_{kj} = \sum_{i=1}^{m} a_{ik} a_{ij}$, $\theta_j = \sum_{i=1}^{m} a_{ij} b_i$, or in compact form

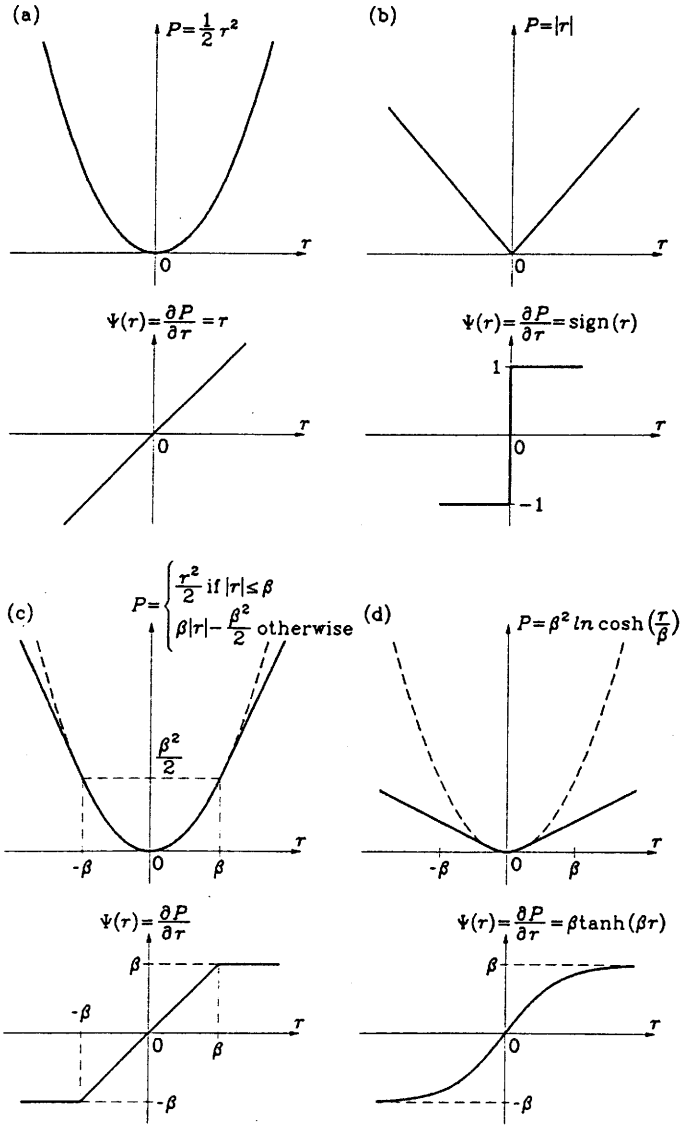


Fig. 2. Plots of examplary penatly functions $P(r)$ and the corresponding activation functions $\Psi(r)$ (see eqn. (23)).

$$\dot{x} = -\mu\Big[\nu\,\varphi(x) + A^T Ax\Big] + \mu A^T b = -\mu\Big[\nu\,\varphi(x) + Wx\Big] + \mu\theta \tag{30b}$$

where $W := A^T A$ and $\theta := A^T b$.

A block diagram illustrating the implementation of this system of differential equations (30a)–(30b) is shown in Fig. 4. This is a Hopfield-type analog neural network with only one layer of processing units (Cichocki and Unbehauen, 1994a).

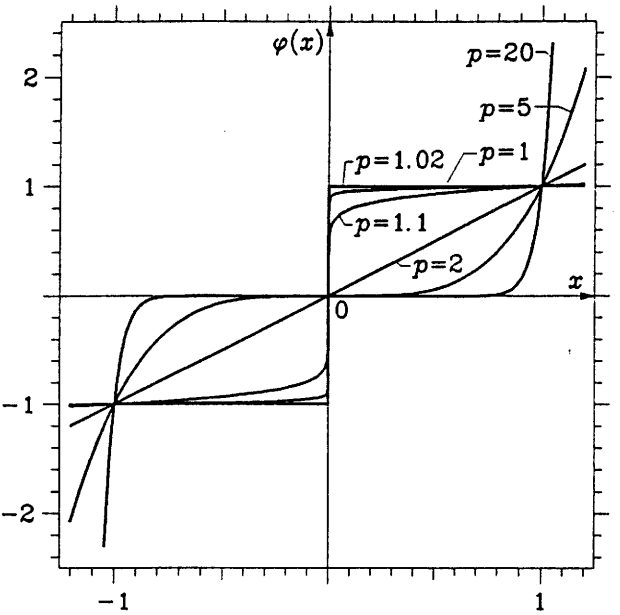


Fig. 3. Examplary plots of the activation function $\varphi(x_i) = \partial f(x)/\partial x_i = |x_i|^{p-1} \operatorname{sign} x_i$ for different values of the parameter p.

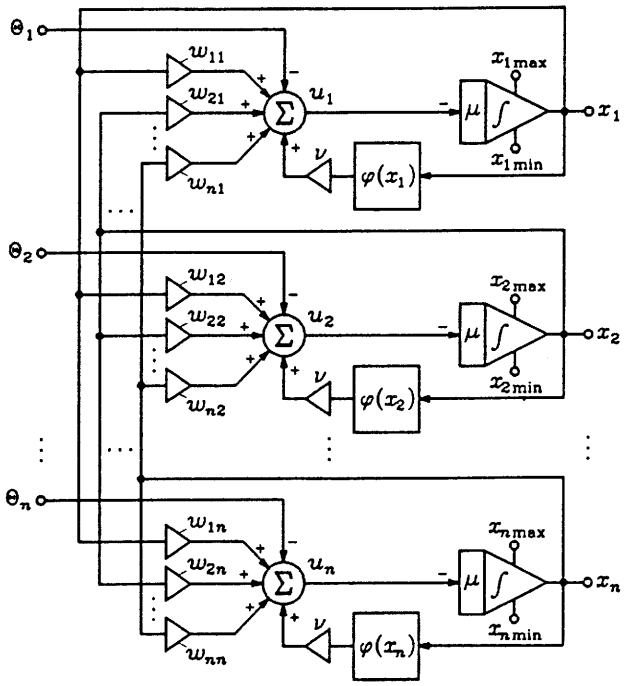


Fig. 4. A simplified neural network for optimization problem (12a)–(12c) (see eqns. (30a)–(30b)). Note that the network requires precomputed values of the biases θ_j and synaptic weights w_{ij}.

The approach described above is very simple and straightforward, however, some problems may arise in practical implementations of the system of differential equations, especially, if the matrix A is very large. Firstly, the VLSI implementation of the neural network architectures shown in Fig. 1 and 4 is a difficult problem because of the complex connectivity between a large number of processing units. The wiring of the large number of processing units on a two-dimensional surface of silicon wafer represents a major bottleneck for VLSI implementations. Secondly, the neural networks proposed in Fig. 1 and Fig. 4 require an extremely large number of programmable weights while the network in Fig. 4 needs mn weights (which can be realized for example as analog four quadrant multipliers). Thirdly, analog VLSI neural circuits are strongly influenced by device mismatches from the fabrication process and a variety of parasitic fabrication processes and consequently a variety of parasitic effects may degrade the final performance. Motivated by the desire to maximally simplify the neural network architecture and alleviate the problems mentioned above, in the next section we will propose a new approach which enables us to design a considerably simplified neural network.

3.2. Simplified Model – Novel Approach

In this section we will limit our considerations to the following optimization problem:

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } Ax = b \end{aligned} \tag{31}$$

and

$$x_{j\,\min} \leq x_j \leq x_{j\,\max}$$

Note that any inequality constraint $a_i^T x \leq b_i$ can be converted to the standard equality form by adding a slack variable, i.e.

$$a_i^T x + x_{n+i} = b_i \quad \forall\, i$$

$$\text{with} \quad x_{n+i} \geq 0$$

To solve optimization problem (31) by an appropriate ANN the key step is to construct a suitable computational energy function $E(x)$. For this purpose we can define instantaneous residuum (or error) function

$$\widetilde{r}\Big(x(t)\Big) := s^T(Ax - b) = \sum_{i=1}^{m} s_i(t) r_i\Big(x(t)\Big) \tag{32}$$

where $r_i(x) = \sum_{j=1}^{n} a_{ij} x_j - b_i$ and $s := \big[s_1(t), s_2(t), \ldots, s_m(t)\big]^T$ is in general the set of zero-mean mutually independent (or uncorrelated) identically distributed (i.i.d.) external excitation signals (e.g. uncorrelated high frequency or pseudorandom deterministic signals). Note that the actual value of the error (residuum) function is equal to zero at any time instant (or during any time period) if and only if the constraint $Ax = b$ is exactly satisfied.

The instantaneous error function $\widetilde{r}\big(x(t)\big)$ can be developed as

$$\begin{aligned}\widetilde{r}\Big(x(t)\Big) &= \sum_{i=1}^{m} s_i(t) r_i\Big(x(t)\Big) = \sum_{i=1}^{m} s_i(t)\Big(\sum_{j=1}^{n} a_{ij} x_j(t) - b_i\Big) \\ &= \sum_{j=1}^{n}\Big(\sum_{i=1}^{m} a_{ij} s_i(t)\Big) x_j(t) - \sum_{i=1}^{m} b_i s_i(t) = \sum_{j=1}^{n} \widetilde{a}_j x_j(t) - \widetilde{b}(t)\end{aligned} \tag{33}$$

where $\widetilde{a}_j(t) := \sum_{i=1}^{m} a_{ij} s_i(t)$, $\widetilde{b}(t) := \sum_{i=1}^{m} b_i s_i(t)$.

For the so-formulated residuum function we can construct the instantaneous estimate of the computational energy (cost) function at the time t as

$$E\Big(x(t)\Big) = \nu f(x) + P\Big[\widetilde{r}\Big(x(t)\Big)\Big] \tag{34}$$

where $\nu > 0$ and $P(\widetilde{r})$ is the penalty function defined by one of the equations (23).

Minimization of the energy function by the gradient descent method leads to the system of differential equations

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = -\mu\Big[\nu\varphi\Big(x_j(t)\Big) + \widetilde{a}_j(t)\Psi\Big[\widetilde{r}\Big(x(t)\Big)\Big]\Big], \qquad j = 1, 2, \ldots, n \tag{35}$$

where $\mu > 0$, $\nu > 0$, $\varphi(x_j) := \dfrac{\partial f(x(t))}{\varphi x_j}$, $\Psi\Big[\widetilde{r}\big(x(t)\big)\Big] := \dfrac{\partial P(\widetilde{r})}{\partial \widetilde{r}}$ and $\widetilde{r}\big(x(t)\big) = \sum_{j=1}^{n} \widetilde{a}_j(t) x_j(t) - \widetilde{b}(t)$.

In the special case of $P(\widetilde{r}) = \dfrac{1}{2}\widetilde{r}^2$ the system of differential equations simplifies to

$$\frac{\mathrm{d}x_j(t)}{\mathrm{d}t} = -\mu\Big[\nu\varphi\Big(x_j(t)\Big) + \widetilde{a}_j(t)\widetilde{r}\Big[\widetilde{r}\Big(x(t)\Big)\Big]\Big], \qquad j = 1, 2, \ldots, n \tag{36}$$

The system of differential equations (35) can be considered as the basic adaptive learning algorithm of a single artificial neuron as shown in Fig. 5 and 6a. The network consists of analog integrators, summers, activation functions and analog four quadrant multipliers (see Figs. 6a,b). The network is driven by the incoming data a_{ij} and b_i multiplied (modulated) by high frequency zero-mean mutually uncorrelated source signals $s_i(t)$. The artificial neuron shown in Fig. 6a with an on chip adaptive learning algorithm allows processing of the information fully simultaneously. If only one signal generator (or a pseudo-random generator) is available in order to generate m excitation signals $s_i(t)$ a chain of unit delays can be employed as shown in Fig. 6b.

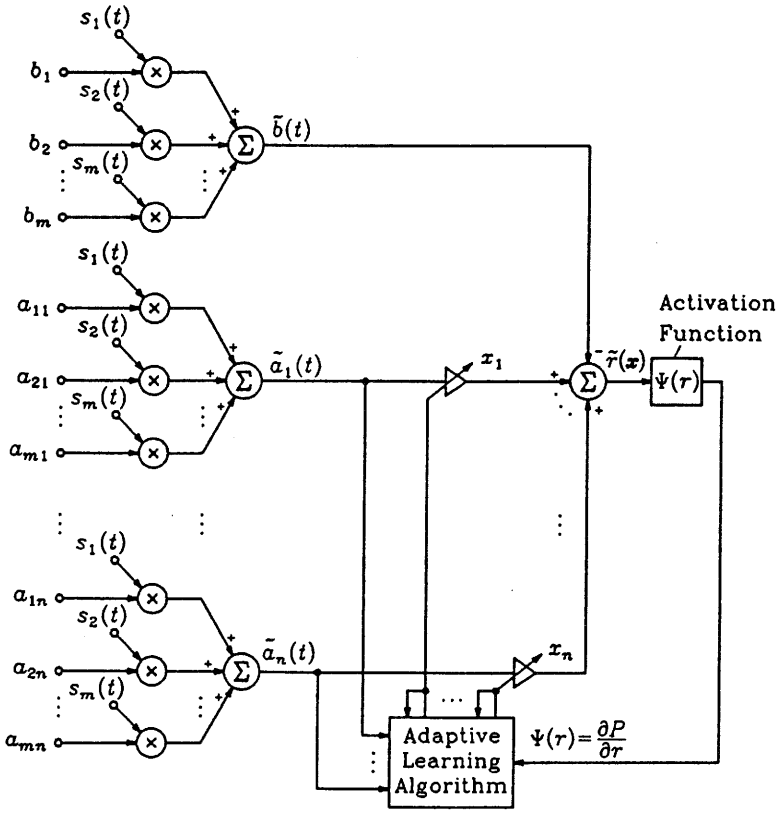


Fig. 5. Functional block diagram of a single artificial neuron with adaptive synaptic weights x_j $(j = 1, 2, \ldots, n)$ and with preprocessed network.

In order to further simplify the neural network implementation we have found that rather expensive analog multipliers can be replaced by the simple CMOS switches S_1 to S_m as shown in Fig. 7a. Various different strategies for controlling the switches can be chosen. In the simplest strategy the switches can be controlled by a multiphase clock, i.e. the switches will be closed and opened cyclically. In this case the network processes the set of equations (constraints) in a cyclical order, i.e. in each clock only one constraint is active. On the other hand, in order to perform a fully simultaneous processing of all the constraints $a_i^T x - b_i = 0$ $(i = 1, 2, \ldots, m)$, the switches S_1 to S_m should be controlled by a digital generator producing multiple pseudo-random, uncorrelated bit streams. As such a generator, for example, a simple feedback shift register can be used which is able to generate uncorrelated multiple mutually shifted pseudorandom bit streams with very good noise-like properties (see Fig. 7b) (Alspector *et al.*, 1991).

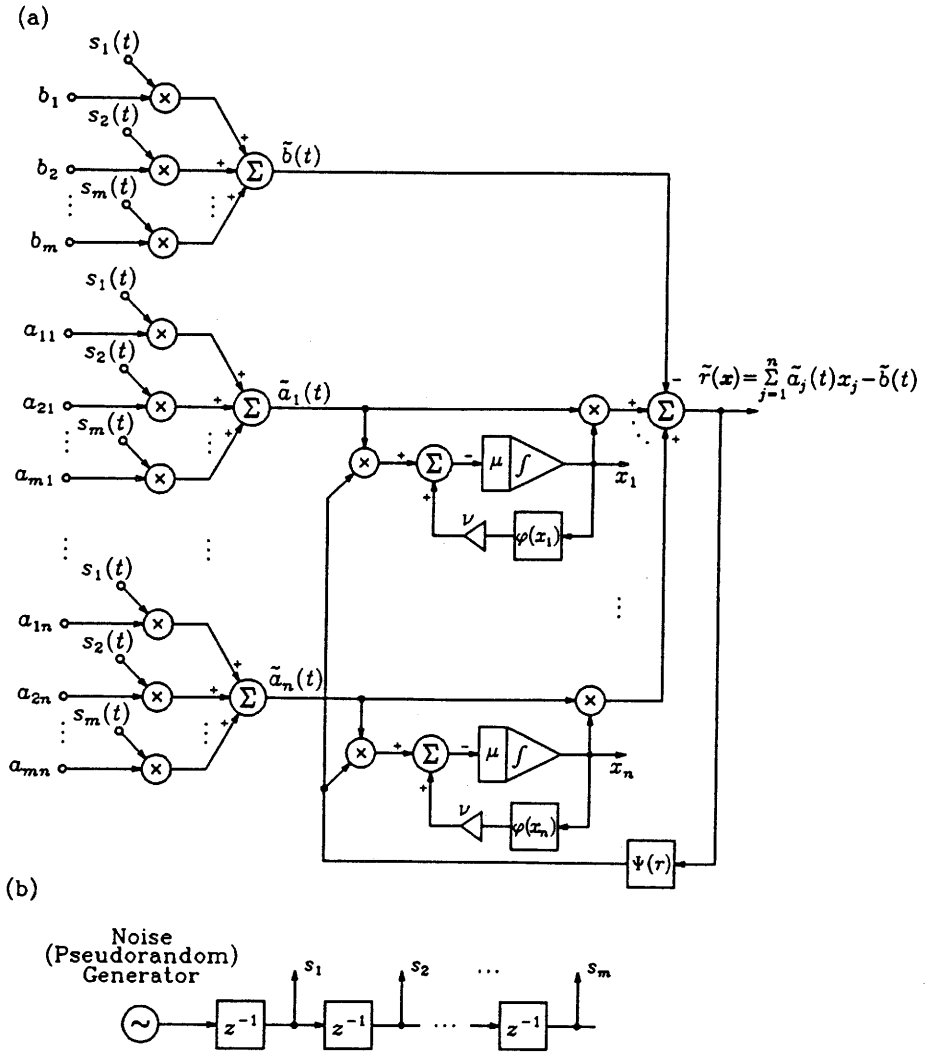


Fig. 6. a) Structure of artificial neuron with continuous-time learning algorithm (36),
b) Exemplary implementation of the system for generating m pseudorandom noise sources $s_i(t)$.

4. Simplified Neural Network Models for TLS Problem

For the total least squares problem formulated in Section 2 we can construct the instantaneous energy function

$$E_{\mathrm{TLS}}(x) = \frac{1}{2}\,\frac{e^2(t)}{x^T x + 1} \tag{37}$$

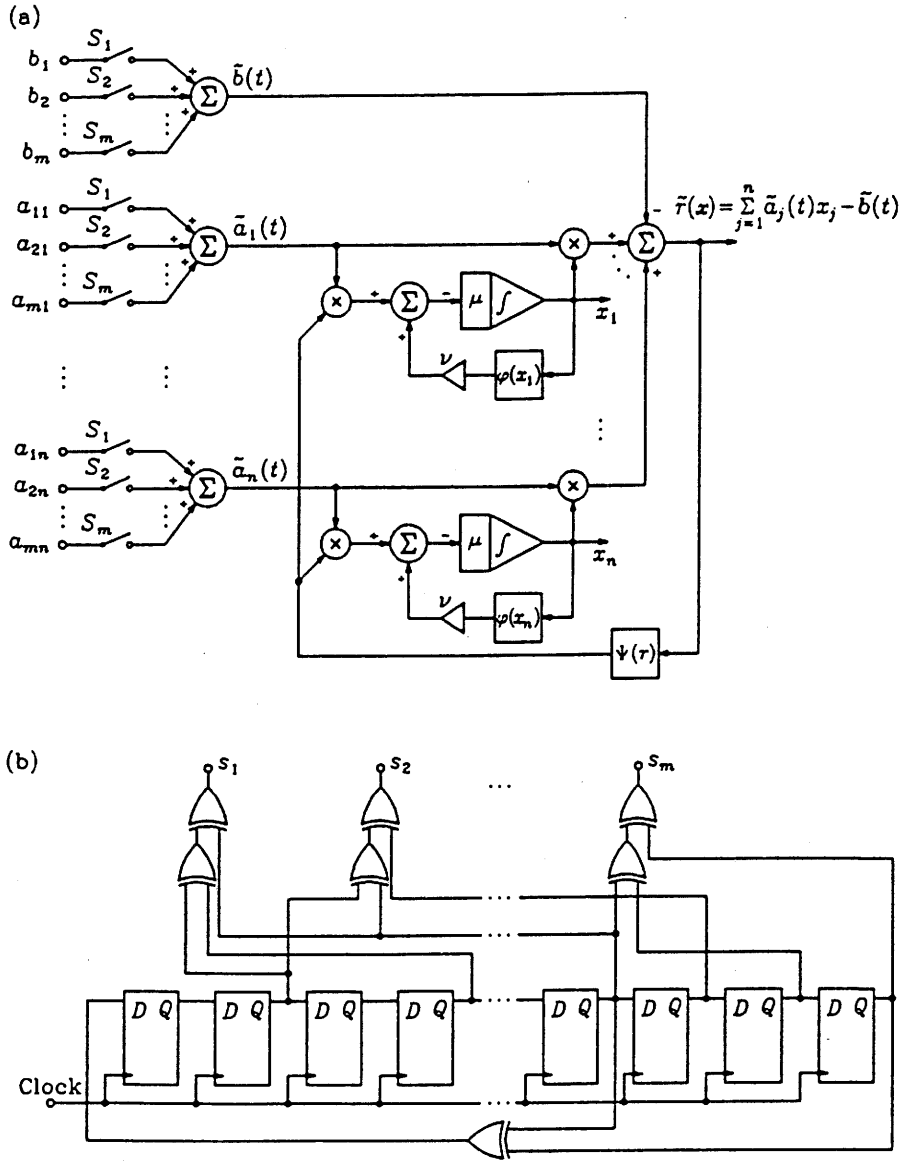


Fig. 7. a) Simplified implementation of the artificial neuron of Fig. 6a,
b) Exemplary realization of a digital circuit generating multiple, mutually uncorreleted, pseudo-random bit streams.

where

$$x = \left[x_1(t), x_2(t), \ldots, x_n(t)\right]^T$$

and

$$e(t) = \widetilde{r}(t) = \sum_{j=1}^{n} \widetilde{a}_j(t)x_j(t) - \widetilde{b}(t) = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij}s_i(t) \right) x_j(t) - \sum_{i=1}^{m} b_i s_i(t)$$

Applying the standard gradient descent algorithm we obtain the set of differential equations

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = -\mu(t)\frac{\partial E_{\mathrm{TLS}}(x)}{\partial x_j} = -\mu(t)e(t)\frac{\widetilde{a}_j(t)(x^T x + 1) - e(t)x_j(t)}{(x^T x + 1)^2} \tag{38}$$

where $\mu(t) > 0$. The above system of differential equations, after linearization, can be simplified as

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = -\mu(t)e(t)\left[\widetilde{a}(t) + \widetilde{b}(t)x(t)\right] \tag{39}$$

The above set of differential equations constitutes a basic adaptive parallel learning algorithm for solving the TLS problem for overdetermined linear systems $Ax \cong b$.

For an ill-conditioned problem the instataneous estimate of the energy function can be formulated as

$$E(x, \nu) = \frac{1}{2}\frac{e^2(t)}{x^T x + 1} + \frac{\nu}{2}\|x\|_2^2 \tag{40}$$

where $\nu \geq 0$ is the standard regularization parameter. In this case the learning algorithm (39) is modified as

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = -\mu(t)\left\{ e(t)\left[\widetilde{a}(t) + \widetilde{b}(t)x(t)\right] + \nu\, x(t) \right\} \tag{41}$$

with $\mu(t) > 0$.

An analog (continuous-time) implementation of algorithm (39) is illustrated in Fig. 8. Note that algorithms (39) and (42) can be considered as a generalization (extension) of the "standard" LS algorithm. Clearly, algorithms (39), (41) can be converted to discrete-time iterative algorithms by applying the Euler rule (Cichocki and Unbehauen, 1994a).

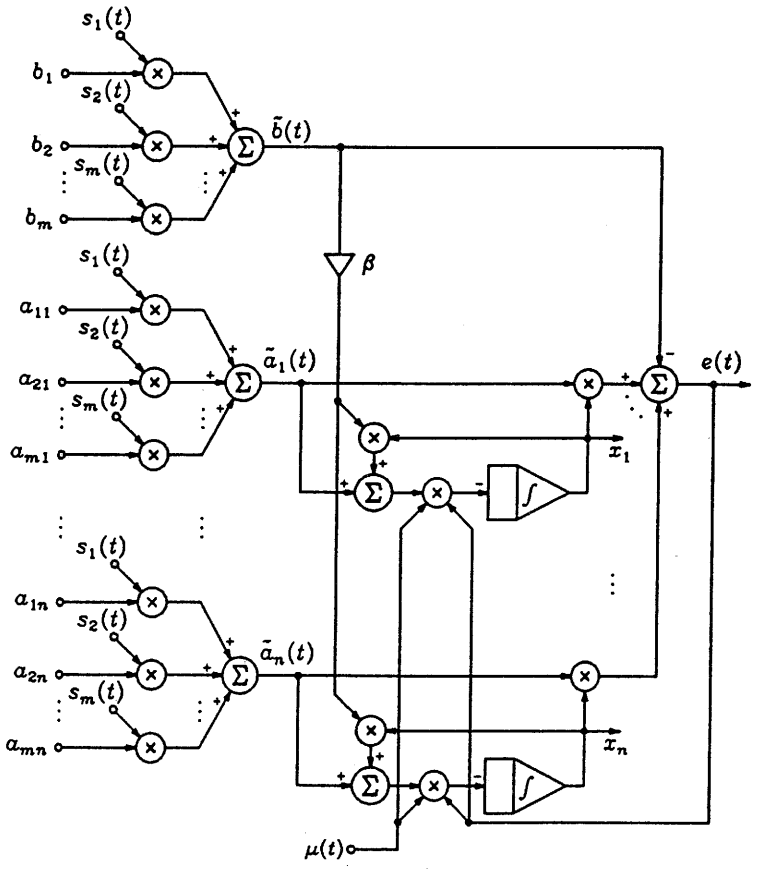


Fig. 8. Architecture of single artificial neuron with continuous-time learning algorithm (39) for the standard LS problem ($\beta = 0$), for the TLS problem ($\beta = 1$) and the DLS problem ($\beta \gg 1$).

Motivated by the desire to increase the maximal convergence speed of the iterative algorithms and inspired by the "averaging concept" proposed recently by Polyak for a recursive formula of stochastic approximation (Polyak, 1990), the following iterative LS/TLS learning algorithm has been proposed (Cichocki and Unbehauen, 1994a; 1994b):

$$\widetilde{x}^{(k+1)} = \widetilde{x}^{(k)} - \frac{\eta_0}{k^{\lambda}} e(k)\left[\widetilde{a}(k) + \beta \widetilde{b}(k)\widetilde{x}^{(k)}\right] \tag{42a}$$

$$x^{(k+1)} = x^{(k)} + \frac{1}{k+1}\left(\widetilde{x}^{(k+1)} - x^{(k)}\right) \tag{42b}$$

where $\eta_0 > 0$, $\frac{1}{2} < \lambda < 1$, $\widetilde{a}(k) = \left[\widetilde{a}_1(k), \widetilde{a}_2, \ldots, \widetilde{a}_n(k)\right]^T$, $k = 0, 1, \ldots,$; and $\beta = 0$ for the LS problem or $\beta = 1$ for the TLS problem. The proposed learning algorithm

combines two processes. The first computing process is a standard stochastic recursive algorithm described by the set of difference equations (32a) with a learning rate $\widetilde{\eta}^{(k)} = \frac{\eta_0}{k^{\lambda}}$, $\left(\frac{1}{2} < \lambda < 1\right)$. The second process is an averaging process described by the set of difference equations (42b) with a learning rate $\eta^{(k)} = 1/(k+1)$. The implementation of the algorithm is illustrated in Fig. 9. The switches are controlled in general by a multiphase pseudo-random generator. The important feature of the learning algorithm (42a)–(42b) is that in contrast to the standard approach two discrete time sequences are constructed where $\{x^{(k)}\}$ is the arithmetic average of $\{\widetilde{x}^{(k)}\}$. Another essential feature of the algorithm is that it employs two learning rates: $\widetilde{\eta}^{(k)} = \eta_0 k^{-\lambda}$ and $\eta^{(k)} = (k+1)^{-1}$ where $\frac{1}{2} < \lambda < 1$. Note that $\lambda = 1$ is excluded from the above algorithm and that the learning rate $\widetilde{\eta}^{(k)}$ decreases more slowly than $\eta^{(k)} = 1/(k+1)$. It should be noted that one may use only the set of eqns. (42a) but at the expense of a slower convergence speed. The use of the second set of equations (cf. eqn. (42b)) improves the convergence speed, i.e. it renders it possible to get the optimal values x^* considerably faster.

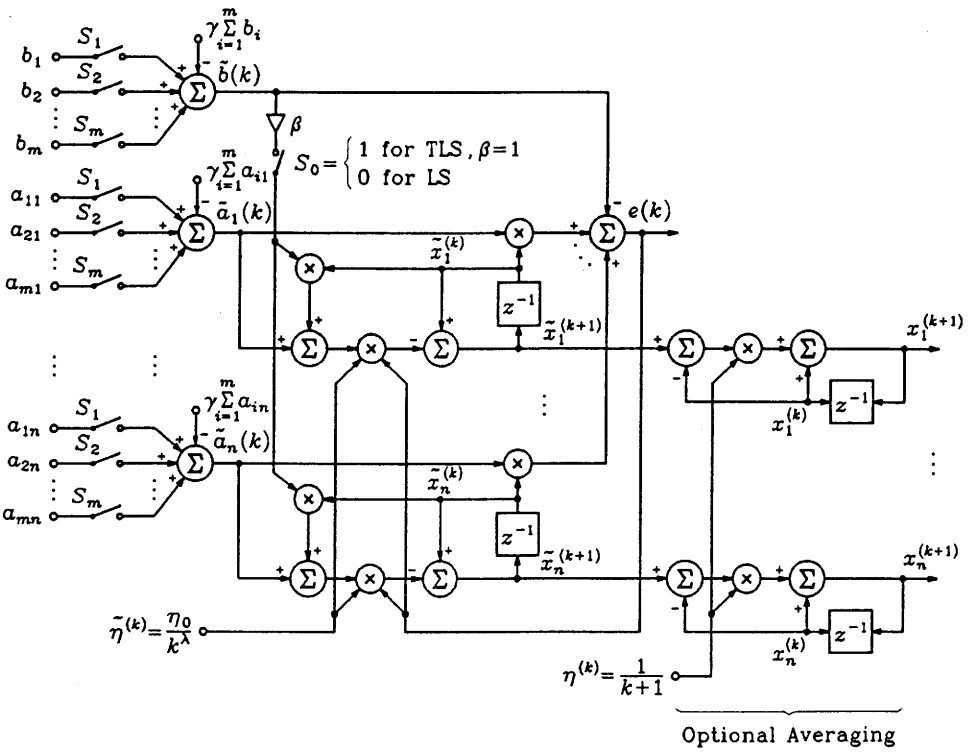


Fig. 9. Functional block diagram illustrating realization of the discrete time (iterative) learning algorithm (42a)–(42b) (For fully simultaneosly operating switches $\gamma = 0.5$, for cylically operating switches $\gamma = 0$).

In practice, the averaging process will start not at $k = 0$ but from $k \geq k_0$ for which $\tilde{x}^{(k)}$ enters the neighbourhood of the desired optimal solution x^*, i.e. the algorithm can take the following special form:

$$\tilde{x}^{(k+1)} = \tilde{x}^{(k)} - \eta \frac{(a_i^T \tilde{x}^{(k)} - b_i)(a_i + \beta b_i \tilde{x}^{(k)})}{a_i^T a_i} \tag{43a}$$

$$x^{(k+1)} = \tilde{x}^{(k)} \qquad \text{for } k = 0, 1, 2, \ldots, k_0, \quad \text{with } 0 < \eta < 2$$

and

$$\tilde{x}^{(k+1)} = \tilde{x}^{(k)} - \frac{\eta_0}{k^\lambda}(a_i^T \tilde{x}^{(k)} - b_i)(a_i + \beta b_i \tilde{x}^{(k)}) \tag{43b}$$

$$x^{(k+1)} = x^{(k)} - \frac{1}{k - k_0}(\tilde{x}^{(k+1)} - x^{(k)}) \quad \text{for } k > k_0,\ \eta_0 > 0,\ \gamma = 0 \tag{43c}$$

A sequence of indices $i = \{i(k)\}_{k=0}^{\infty}$ according to which the rows of the matrix A and the elements of the vector b are taken up is called a control strategy. Clearly, various different strategies for controlling the switches $\{S_i\}$ can be chosen (cf. Fig. 7). In the simplest control strategy we have $i = k \,(\mathrm{mod}\, m) + 1$; i.e. the switches are closed and opened cyclically. In this case the switches can be controlled by a simple multiphase clock generator. The important problem of choosing an optimal control strategy is out of the scope of this paper and is left for future investigations.

The proposed algorithm (43) has the following features:

i) No operations are performed on a linear system $Ax \cong b$ as a whole,

ii) Each iterative step requires the access to only one row of the matrix A and b,

iii) The fact that the algorithm needs only one row at a single iterative step makes this scheme an especially useful tool for solving large unstructured systems according to the LS and/or TLS criterion,

iv) As shown by some computer experiments the algorithm allows us to increase the rate of convergence in comparison to the "standard" iterative algorithm without averaging.

Algorithm (43) can be considered as an extension (generalization) of the Kaczmarz algorithm.

5. Further Extensions and Generalizations of Neural Network Models

It is interesting that the neural network models shown in Figs. 8–9 can be employed not only to solve LS or TLS problems but they can easily be modified and/or extended to related problems: e.g. linear and quadratic programming, iteratively reweighted LS problems (Cichocki and Unbehauen, 1992), minimum norm $(1 \leq p \leq \infty)$ and/or maximum entropy problems (Censor *et al.*, 1989).

Furthermore, by changing the value of the parameter β (cf. Figs. 8,9) more or less emphasis can be given to errors of the matrix A with respect to errors of the

vector b. In an extreme case, for large β (say $\beta = 100$) it can be assumed that the vector b is almost free of error and the error lies in the data matrix A only. Such a case is referred to as the so called DLS (data least squares) problem (since the error occurs in A but not in b) (DeGroat and Dowling, 1993).

The DLS problem can be formulated as an optimization problem of finding an optimal vector $x^* = x^*_{\mathrm{DLS}}$ which satisfies the overdetermined system of linear equations

$$(A + \Delta A)x^*_{\mathrm{DLS}} = b \tag{44}$$

under the condition that the Frobenius norm of the error matrix $\|\Delta A\|_F$ is minimal. It can easily be demonstrated that the neural networks of Figs. 8 and 9 solve approximately the DLS problem for a large β (typically $\beta = 50$ to 100, see Example 3). In other words, the DLS problem can be solved by simulating the system of differential equations

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = -\mu(t)e(t)\left[\widetilde{a}(t) + \beta\widetilde{b}(t)x(t)\right] \tag{45a}$$

with $\mu(t) > 0$, $\beta \gg 1$.

For complex-valued elements (signals) the algorithm can further be generalized as

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = -\mu(t)e(t)\left[\widetilde{a}^c(t) + \beta\widetilde{b}^c(t)x(t)\right] \tag{45b}$$

where the superscript c denotes the complex-conjugate operation, and $\beta = 0$ for the LS-problem, $\beta = 1$ for the TLS problem or $\beta \gg 1$ for the DLS problem. Very recently, it has been shown that the DLS estimation is more appropriate than TLS and ordinary LS for certain types of signal processing problems (DeGroat and Dowling, 1993).

6. Computer Simulation Results

To check to correctness and performance of the proposed algorithms and associated neural network structures we have simulated them extensively on a computer. The networks shown in Figs. 6–9, 1 have been investigated for many different numerical examples and a good agreement with the theoretical considerations has been obtained. Due to limited space in this paper we shall present only some illustrative examples.

Example 1. Consider the problem of finding the minimal 2-norm solution of the underdetermined system of linear equations (Lillo *et al.*, 1993)

$$Ax = b$$

with

$$A = \begin{bmatrix} 2 & -1 & 4 & 0 & 3 & 1 \\ 5 & 1 & -3 & 1 & 2 & 0 \\ 1 & -2 & 1 & -5 & -1 & 4 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ 1 \\ -4 \end{bmatrix}$$

Clearly the above set of equations has infinitly many solutions. However, there is a unique minimum norm solution which we want to find. On the basis of the general architectures shown in Fig. 6 corresponding continuous-time (analog) circuits have been designed and simulated on a computer. For the network of Fig. 6 as excitation signals $s_i(t)$ $(i = 1, 2, 3)$ pseudo-random, independent signals over the range $[-1, 1]$ were used.

For the network of Fig. 7 the switches were controlled by a three phase clock with clock frequency $f_c = 100\,\text{MHz}$. Exemplary computer simulation results are shown in Fig. 10. The final solution (equilibrium point) was

$$x^* = \left[0.0882,\ 0.1083,\ 0.2733,\ 0.5047,\ 0.3828,\ -0.3097\right]^T$$

which is in excellent agreement with the exact minimum 2-norm solution obtained by using MATLAB. Note that the network of Fig. 6 reaches the solution in a time less than 100 nanoseconds. The implementation of the network architecture shown in Fig. 8 leads to the same results.

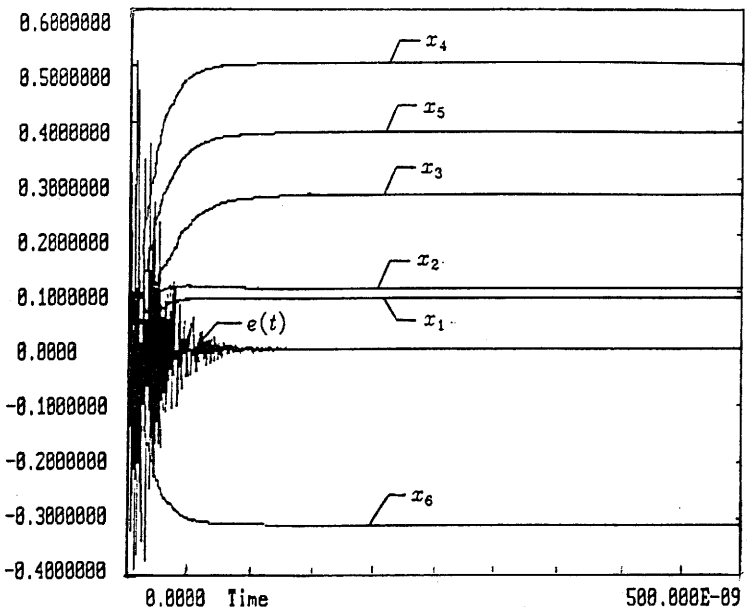


Fig. 10. Computer simulated trajectories $x_j(t)$ and the energy function $E(t)$ for Example 1.

Example 2. Find the pseudodoinverse of the singular matrix

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

In order to find the pseudoinverse matrix $B = A^+$, we need to make the observation vector b successively $[1, 0, 0]^T$, $[0, 1, 0]^T$ and $[0, 0, 1]^T$. We have repeated the computation three times for each observation vector b to find three columns of the pseudoinverted matrix B. Fig. 11 shows the exemplary computer simulated state trajectories $b_{ij}(t)$ $(i, j = 1, 2, 3)$ by employing the network architecture shown in Fig. 8. The network was able to find

$$B = A^+ = \begin{bmatrix} -0.6387 & -0.1663 & 0.3055 \\ -0.0554 & -0.0001 & 0.0552 \\ 0.5277 & 0.1666 & -0.1942 \end{bmatrix}$$

in the total time less thank 8 μs. The pseudoinverse matrix found by MATLAB reads

$$A^+_{\text{MATLAB}} = \begin{bmatrix} -0.6389 & -0.1667 & 0.3056 \\ -0.0556 & -0.0000 & 0.0556 \\ 0.5278 & 0.1667 & -0.1944 \end{bmatrix}$$

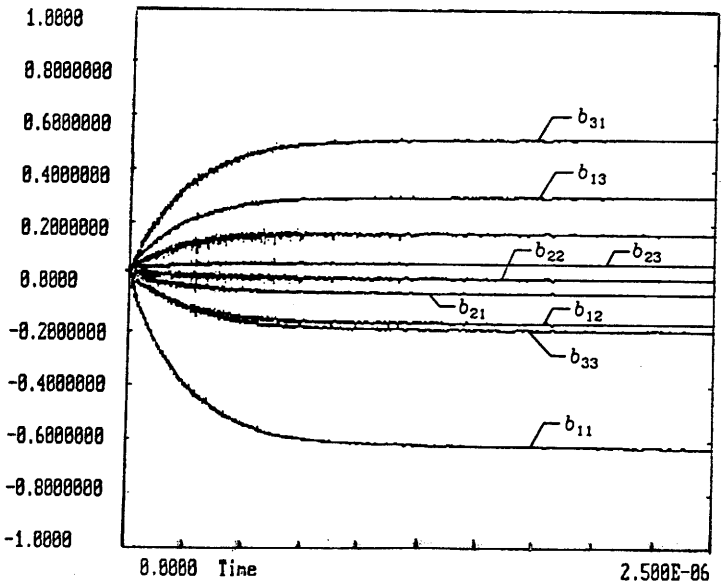


Fig. 11. Computer simulated trajectories $b_{ij}(t)$ for Example 2.

Example 3. Let us consider the following linear parameter estimation problem described by the set of linear equations (Cichocki and Unbehauen, 1994b)

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 2 \end{bmatrix}$$

It is required to find an optimal solution according to the LS, TLS and DLS criterion.

To solve the problem we have employed neural network architectures depicted in Figs. 8 and 9. Exemplary computer simulated results are shown in Fig. 12a and Fig. 12b. It has been found that the analog (fully simultaneous) network of Fig. 8 shows better perfomance (higheconvergence speed) in comparison to the discrete-time (one equation per iterative step) network of Fig. 9. For example the network of Fig. 8 was able to find, in a time less than 400 nanoseconds, the following parameters

$$x^*_{\mathrm{LS}} = [x^*_1, x^*_2]^T = [0.706, 0.298]^T, \qquad \text{for } \beta = 0$$

$$x^*_{\mathrm{TLS}} = [x^*_1, x^*_2]^T = [0.8531, 0.2591]^T, \qquad \text{for } \beta = 1$$

$$x^*_{\mathrm{DLS}} = [x^*_1, x^*_2]^T = [1.1132, 0.1898]^T, \qquad \text{for } \beta = 50$$

which are in good agreement with the exact results obtained by using MATLAB:

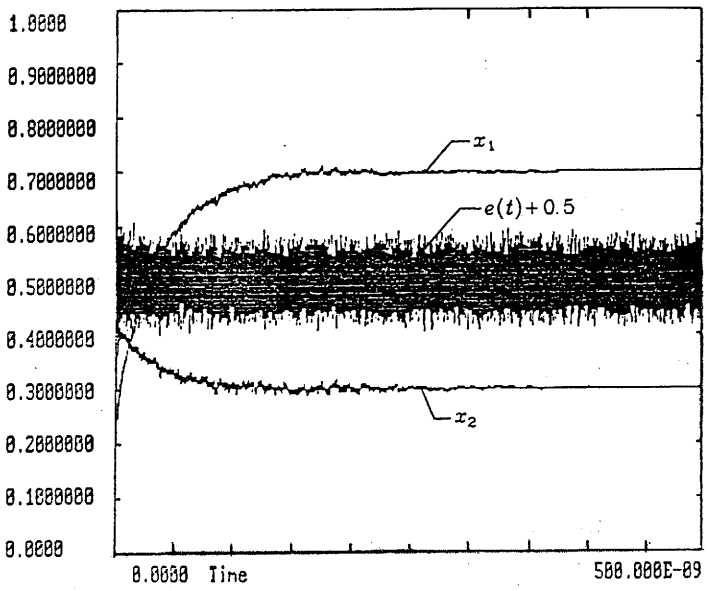
$$x^*_{\mathrm{LS\,MATLAB}} = [0.7,\ 0.3]^T$$

$$x^*_{\mathrm{TLS\,MATLAB}} = [0.8544,\ 0.2587]^T$$

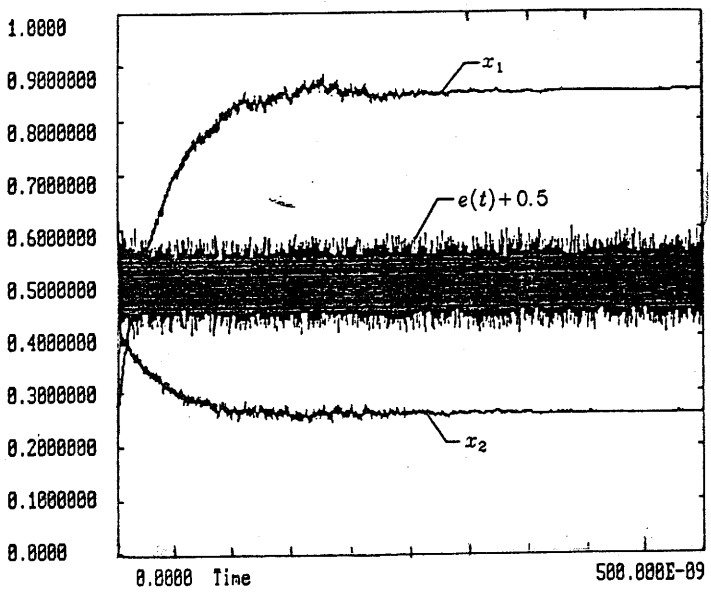
$$x^*_{\mathrm{DLS\,MATLAB}} = [1.1200,\ 0.1867]^T$$

Example 4. In order to test the performance of the proposed neural network for a large number of variables we have simulated it for a tomographic image reconstruction problem from projection (Censor *et al.*, 1989; Frieden and Zoltani, 1985; Kak and Slaney, 1987; Lu *et al.*, 1992; Madych, 1991; Wang and Lu, 1992). In the computer simulation experiments a modification of the Shepp and Logan head phantom has been used as an original image. The Shepp and Logan phantom is often used for testing different numerical algorithms for image reconstruction because it approximately represents a cross-section of the human head which is known to place the greatest demands on a tomographic system with respect to its accuracy (Censor *et al.*, 1989; Frieden and Zoltani, 1985; Lu *et al.*, 1992; Madych, 1991; Wang and Lu, 1992).

Consider a 2-D discrete model for the $n \times n$ cross-section image. Let x_j $(j = 1, 2, \ldots, n)$ denote the density (gray) level of the j-th pixel and $b_i = \sum_{j=1}^{n} a_{ij} x_j$ $(i = 1, 2, \ldots, m)$ represent the i-th ray and j-th pixel. Thus the reconstruction of images from projection can be formulated as a problem of solving a linear system of equations: $Ax = b$, where $A \in \mathbb{R}^{m \times n}$ is the projection matrix, $b \in \mathbb{R}^m$ is the projection data (measurement) vector and $x \in \mathbb{R}^n$ is the density, while n is the number of pixels and m is the number of measurements. Thus the goal is to find or estimate a density vector x assuming that the projection matrix A and data vector b are known. Using the Shepp and Logan head phantom model, we analytically determined the projection matrix $A \in \mathbb{R}^{m \times n}$ for the test image with 64 rays in each of the 56 projections; and the image is reconstructed on a 64×64 sampling lattice. Note that the involved system of linear equations is underdetermined, i.e. $n = 4096$ variables (pixels) and $m = 3584$ equations have been used.



a)



b)

Fig. 12. Exemplary simulated trajectories for Example 3
a) for the LS problem ($\beta = 0$),
b) for the TLS problem ($\beta = 1$).

The image reconstruction has been simulated by using the neural network architecture shown in Fig. 7. The parameters of the network have been chosen as follows: $\mu(t) = 10^6 = \text{const}$, $\nu = 0.02$ all switches were controlled by a multiphase pseudo-random binary generator with a basic clock frequency $f_c = 100\,\text{MHz} = 10^8\,\text{Hz}$. The exemplary images reconstructed from the same projection data, starting from zero initial conditions at the time $1\,\mu\text{s}$, $2\,\mu\text{s}$, $5\,\mu\text{s}$ and $10\,\mu\text{s}$ are shown in Figs. 13–15. The image quality has been evaluated by using the normalized root mean squared (NRMS) error defined as

$$\varepsilon = \left[\frac{\sum_{j=1}^{n} (x_j - x_j^*)^2}{\sum_{j=1}^{n} (x_j - \bar{x})^2} \right]^{\frac{1}{2}}$$

where x_j^* is the reconstructed image at the j-th pixel, x_j is the original image, $\bar{x}$ is the mean of the original image and n is the total number of pixels. In our computer experiments we obtain e.g. the NRMS error $\varepsilon = 0.0547$ for 1-norm criterium after a time less that $10\,\mu\text{s}$, which means that the reconstructed image was a quite good replica of the original image. It is also interesting to note that the computation time (i.e. the settling time in our experiments less than 10 mircoseconds) was principally not influenced by the size of the problem (in contrast to the standard digital computers) but rather depends on the values of the parameters; the learning rate $\mu(t)$, the regularization parameter $\nu(t)$ and the clock frequency f_c.

7. Conclusions

New, very simple and low-cost analog neural networks for solving least squares and total least squares problems have been proposed. In fact, such problems can be solved by using only one single highly simplified artificial neuron with an on chip learning capability. The proposed networks are able to estimate the unknown parameters in real time, i.e. in a time of the order of hundreds or thousands of nanoseconds. The network architectures are suitable for currently available VLSI implementations. The potential usefulness of the proposed networks may be atractive for real time and/or high throughput rate applications, e.g. in robotics, computed tomography and automatic control when the entries of the observation vector and the model matrix are changing in time and it is necessary to continuously track or update the solutions. An interesting and important feature of the proposed algorithmic scheme is its universality and flexibility. It allows either the processing of all equations fully simultaneously in time (cf. Figs. 5,6 and Fig. 8) or processing of groups of equations (i.e. blocks) in every iterative step (cf. Figs. 7,9). These blocks need not be fixed but may rather vary dynamically throughout the iterations. In a special case it allows the processing only of one equation per block, i.e. in each iterative step only one single equation can be processed. We believe that the fully simultaneous algorithmic scheme opens new, and till now not fully explored, vistas in many areas in which the problem arises to solve large systems of linear equations in real time.

Head Phantom 64 x 64;

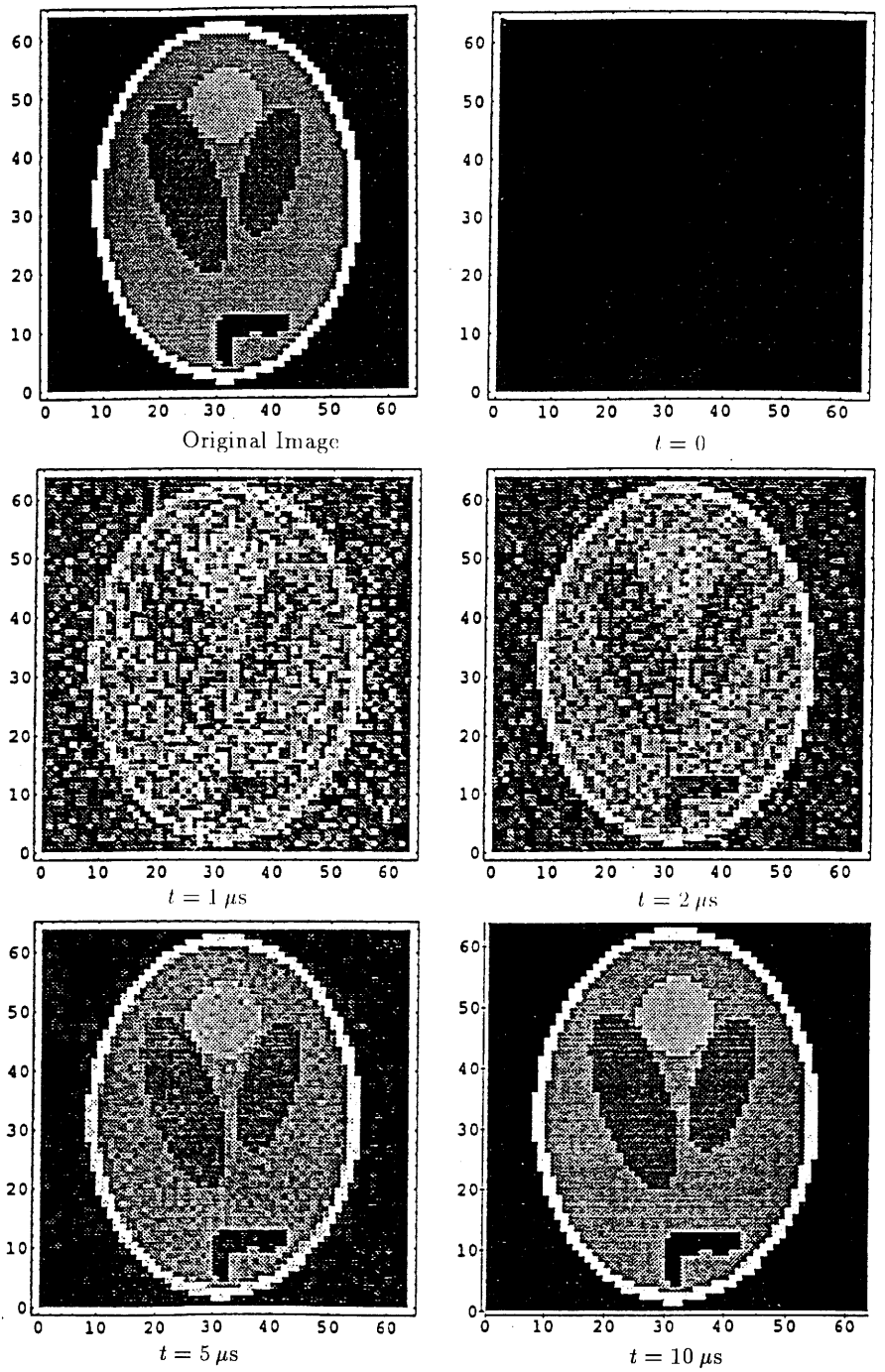


Fig. 13. Original and reconstructed 64 x 64 head-phantom starting with zero initial conditions at the time 0, 1 μs, 2 μs, 5 μs and 10 μs, using 1-norm criterion.

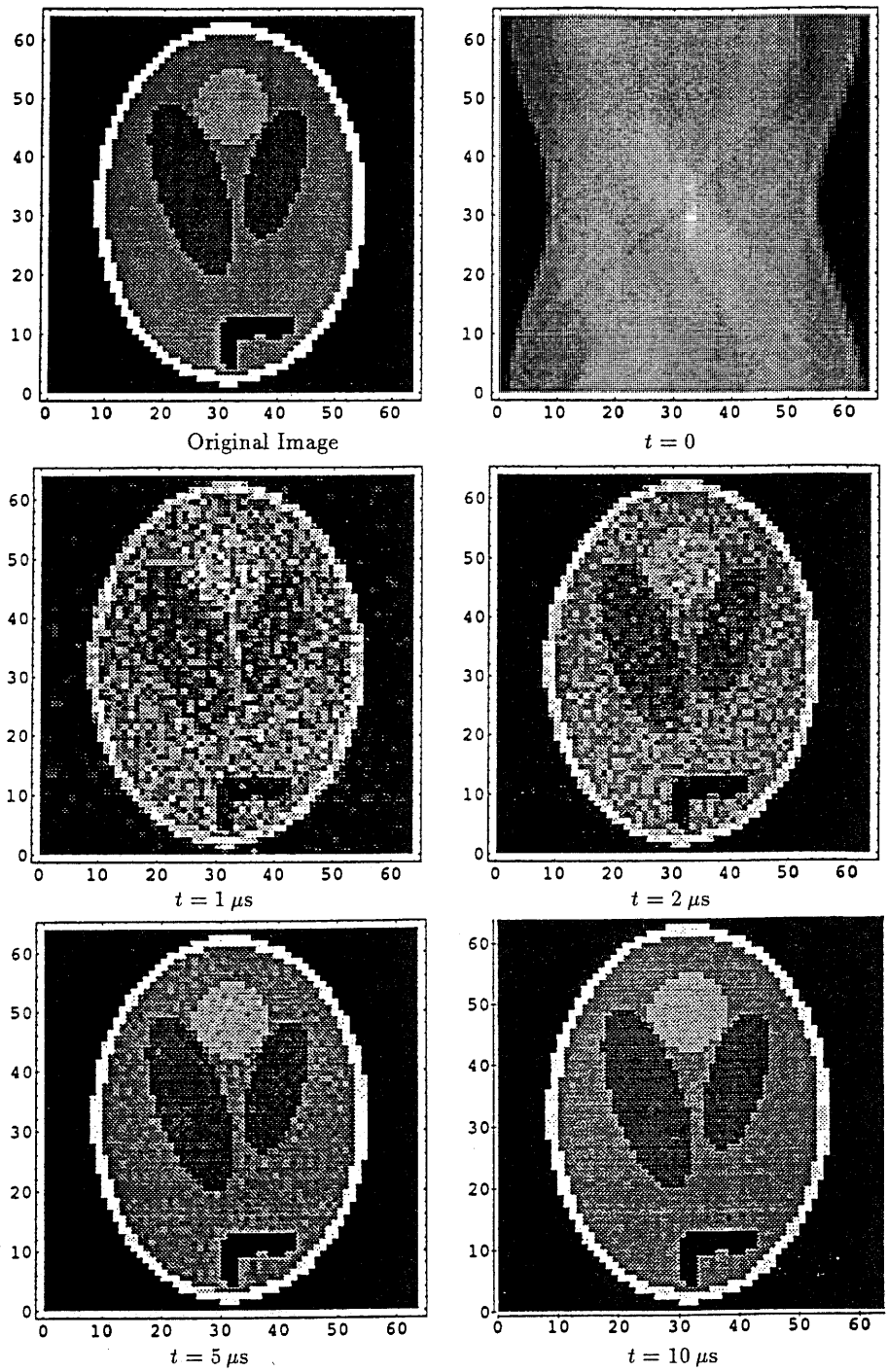
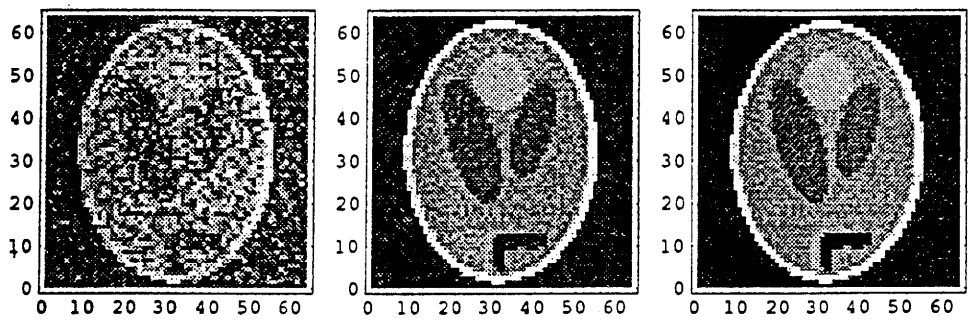
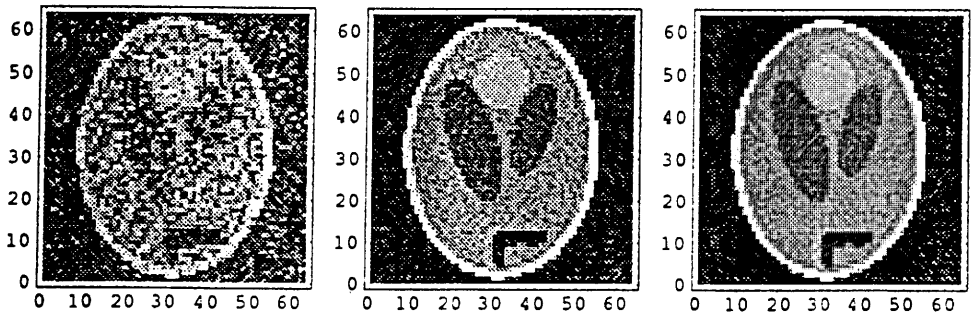


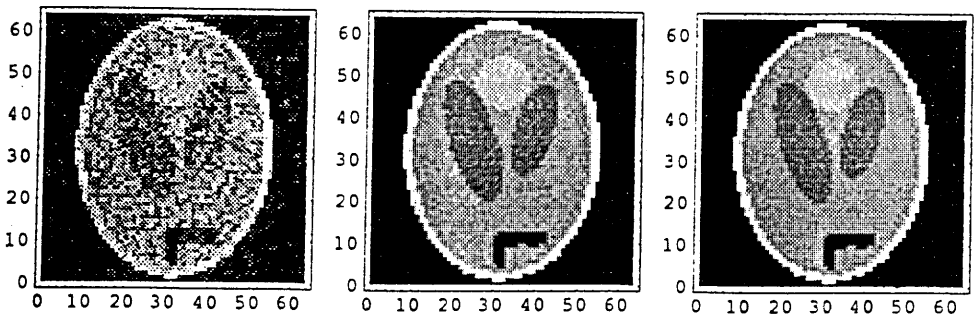
Fig. 14. Original and reconstructed head-phantom starting from non-zero initial conditions at the time 0, 1 μs, 2 μs, 5 μs and 10 μs, using maximum entropy criterion.



a) 1-norm 1 μs / 5 μs / 10 μs



b) 2-norm 1 μs / 5 μs / 10 μs



c) Maximum entropy: 1 μs / 5 μs / 10 μs

Fig. 15. Reconstructed head-phantom images 64 x 64 starting from zero initial conditions at the time 1 μs, 5 μs and 10 μs, using:
a) the minimum 1-norm critrion ($\varepsilon_1 = 0.0476$),
b) the minimum 2-norm critrion ($\varepsilon_2 = 0.0635$),
c) the maximum Shanon entropy critrion ($\varepsilon_{\mathrm{ME}} = 0.0297$).

References

Alspector J., Gannett J.W., Haber S., Parker M.B. and Chu R. (1991): *A VLSI efficient technique for generating multiple uncorrelated noise source and its application to stochastic neural networks.* — IEEE Trans. Circuits Syst., v.38, No.1, pp.109–123.

Cadzow J.A. (1973): *Functional analysis of the optimal control of linear discrete systems.* — Int. J. Control, v.17, No.3, pp.481–495.

Carpenter G. and Grossberg S. (1987): *A massively parallel architecture for a self-organizing neural patterns recognition machine.* — Computer Visin, Graphics and Image Processing, v.37, pp.54–115.

Censor Y., De Pierro A.R., Elfving T., Herman G.T. and Iusen A.N. (1989): *On iterative methods for lineary constrained entropy maximization.* — In: A. Wakulicz (Ed.) Numerical Analysis and Mathematical Modelling, Warsaw: Banach Center Publication, v.XXIV, pp.147–165.

Cichocki A. and Kaczorek T. (1992a): *Neural-type structured networks for solving algebraic Ricatti equations.* — Archives of Control Sciences, v.1, pp.153–165.

Cichocki A. and Kaczorek T. (1992b): *Solving algebraic matrix equations by the use of neural networks.* — Bulletin of the Polish Academy of Sciences, v.40, pp.61–68.

Cichocki A. and Unbehauen R. (1992): *Neural networks for solving systems of linear equations and related problems.* — IEEE Trans. Circuits and Systems, v.39, pp.124–128.

Cichocki A. and Unbehauen R. (1994a): *Neural Networks for Optimization and Signal Processing.* — Chichester: J. Wiley.

Cichocki A. and Unbehauen R. (1994b): *Simplified neural networks for LS and TLS problems in real-time.* — IEEE Trans. Neural Networks, v.5, pp.910–923.

Cichocki A., Ramirez-Angelo J. and Unbehauen R. (1992): *Architectures for analog VLSI implementation of neural networks for solving linear equations with inequality constraints.* — Proc. IEEE Int. Symp. *Circuits and Systems*, pp.1216–1222.

Cichocki A., Kaczorek T. and Stajniak A. (1992): *Computation of the Drazin inverse of a singular matrix by the use of neural network.* — Bull. of the Polish Academy of Science, v.40, pp.386–394.

Cichocki A., Unbehauen R., Lendl M. and Weinzierl K. (1995): *Neural networks for problems with incomplete data especially in applications to signal and image reconstruction.* — Neurocomputing (in print).

DeGroat R.D. and Dowling E.M. (1993): *The data least squares problem and channel equalization.* — IEEE Trans. Signal Processing, v.41, pp.407–411.

Frieden B.R. and Zoltani C.R. (1985): *Maximum bounded entropy: Application to tomographic reconstruction.* — Appl. Opt., v.24, No.2, pp.201–297.

Golub G.H. and Van Loan C.F. (1980): *An analysis of the total least squares problem.* — SIAM J. Numer. Anal., v.17, pp.883–893.

Golub G.H. and Van Loan C.F. (1989): *Matrix Computation.* — Baltimore: Johns Hopkins University Press.

Grossberg S. (1988): *Non-linear neural networks principles, mechanisms and architectures.* — Neural Networks, v.1, pp.17–61.

Herman G.T. (1980): *Image Reconstruction from Projections.* — New York: Academic Press.

Herman G.T., Louis A.K. and Natterer F. (Eds.) (1991): *Mathematical Methods in Tomography.* — Berlin: Springer Verlag.

Hopfield J.J. (1984): *Neurons with graded response have collective computation properties like those of two-state neurons.* — Proc. Nat. Acad. Sci. U.S., v.81, pp.3088–3092.

Ingman D. and Merlis Y. (1992): *Maximum entropy signal reconstruction with neural networks.* — IEEE Trans. Naural Networks, v.3, pp.195–201.

Kaczmarz S. (1937): *Angenaehrte Aufloesung von Systemen linearen Gleichungen.* — Bull. Acad. Pol. Sci., pp.355–357, Left A.

Kak A.C. and Slaney M. (1987): *Principles of Computerized Tomographic Imaging.* — New York: IEEE Press.

Kennedy M.P. and Chua L.O. (1988): *Neural networks for linear programming.* — IEEE Trans. Circuits Syst., v.CAS–35, pp.554–564.

LaSalle J.P. (1986): *The stability and control of discrete processes.* — New York: Springer Verlag.

Levy S. and Fullagar P.K. (1981): *Reconstruction of a sparse spike train from a portion of its amplitude spectrum and application to high resolution deconvolution.* — Geophysics, v.46, pp.1235–1243.

Lillo W.E., Hui S. and Żak S.H. (1993): *Neural networks for constrained optimization problems.* — Int. J. Circuit Theory and Applications, v.21, No.4, pp.385–399.

Ljung L. (1977): *Analysis of recursive stochastic algorithms.* — IEEE Trans. Automatic Control, v.22, pp.551–575.

Lu T., Udpa S.S. and Udpa L. (1992): *Projection iterative reconstruction technique and optoelectronic implementation.* — Proc. ISCAS 92, pp.2469–2472.

Madych W.R. (1991): *On underdetermined systems.* — In: Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, (Eds.): G.H. Golub and P. Van Dooren, Berlin: Springer Verlag, pp.541–545.

Mammone R.J. (Ed.) (1992): *Computational Methods of Signals Recovery and Recognition.* — New York: J. Wiley.

Marrian C.R.K. and Peckerar M.C. (1989): *Electronic "neural" net algorithm for maximum entropy solution of ill-posed problems.* — IEEE Trans. Circuits Syst., v.36, pp.209–219.

Martinelli G., Orlando G. and Burrascano P. (1986): *Spectral estimation by repetitive L_1-norm minimization.* — Proc. IEEE, v.74, pp.523–524.

O'Leary D.P. (1990): *Robust regression computation using iteratively reweighted least squares.* — SIAM J. Matrix Anal. Appl., v.11, pp.466–480.

Osowski S. (1993): *Neural networks in interpolation problems.* — Neurocomputing, v.5, pp.105–118.

Polyak B.T. (1990): *New method of stochastic approximation type.* — Automat. Remote Contr., v.51, pp.937–947.

Rosenfield A. and Kak A.C. (1981): *Digital Picture Processing.* — New York: Academic Press, v.1.

Smith M.J.S. and Portmann C.L. (1989): *Practical design analysis of a simple "neural" optimization circuit.* — IEEE Trans. Circuits Syst., v.36, pp.42–50.

Takefuji Y. (1992): *Neural Network Parallel Computing.* — Boston: Kluwer.

Tank D.W. and Hopfield J.J. (1986): *Simple "neural" optimization networks: an A/D converter, signal decision circuits, and a linear programming circuits.* — IEEE Trans. Circuits Syst., v.CAS–33, pp.533–541.

Unbehauen R. and Cichocki A. (1989): *MOS Switched-Capacitor and Continuous-Time Integrated Circuits and Systems.* — New York: Springer Verlag.

Van Huffel S. and Vandewalle J. (1989): *Algebraic connections between the least squares and total least squares problems.* — Numer. Math., v.55, pp.431–449.

Van Huffel S. and Vandewalle J. (1991): *The Total Least Squares Problem.* — Computational Aspects and Analysis, Frontiers in Applied Mathematics, v.9, Philadelphia: Society for Industrial and Applied Mathematics.

Wang Y. and Lu W. (1992): *Multicriterion maximum entropy image reconstruction from projection.* — IEEE Trans. Medical Imaging, v.11, No.1, pp.70–75.

Widrow B. and Lehr M.A. (1990): *30 years of adaptive neural networks: Perceptron, Madaline and backpropagation.* — Proc. IEEE, v.78, pp.1415–1442.

Yin G. and Zhu Y. (1992): *Averaging procedures in adaptive filtering: an efficient approach.* — IEEE Trans. Automatic Control, v.37, pp.466–475.

Zala C.A., Barrodale I. and Kennedy J.S. (1985): *Comparison of algorithms for high resolution deconvolution of array beamforming output.* — In: Acoustical Imaging, v.14 (Eds.): A.J. Berkhout, J. Ridder and L.F. van der Wal, New York: Plenum Press, pp.699–702.

Zhou Y.T., Chellappa R. and Jenkins B.K. (1988): *Image restoration using a neural net.* — IEEE Trans. Acoustics, Speech, and Signal Processing, v.36, pp.1141–1152.

Received: December 10, 1994