Neural Networks for Solving Systems of Linear Equations and Related Problems

Andrzej Cichocki and Rolf Unbehauen, *Fellow, IEEE*

Abstract—**In this paper various circuit architectures of simple neuron-like analog processors are considered for on-line solving of a system of linear equations with real constant and/or time-variable coefficients. The proposed circuit structures can be used, after slight modifications, in related problems, namely, inversion and pseudo-inversion of matrices and for solving linear and quadratic programming problems. Various ordinary differential equation formulation schemes (generally nonlinear) and corresponding circuit architectures are investigated to find which are best suited for VLSI implementations. Special emphasis is given to ill-conditioned problems. The properties and performance of the proposed circuit structures are investigated by extensive computer simulations.**

I. Introduction

SOLVING systems of linear equations is considered to be one of the basic problems widely encountered in science and engineering since it is frequently used in many applications [7]–[12]. For example, every linear parameter estimation problem gives rise to an overdetermined set of linear equations $Ax \cong b$. This problem arises in a broad class of scientific disciplines such as signal processing, robotics, automatic control, system theory, statistics, and physics [17]. In many applications an on-line (i.e., in real-time) solution of a set of linear equations (or, equivalently, an on-line inversion of matrices) is desired. For such real-time applications when the solution is to be obtained within a time of the order of a hundred nanoseconds, a digital computer often cannot comply with the desired computation time, or its use is too expensive. There are two possible approaches to solve this problem. One approach is to use systolic or wavefront arrays [8]. Another approach is to employ artificial neural networks (ANN's) [2]–[6], [10] which can be considered specialized analog computers relying on strongly simplified models of neurons. The recent wave of interest in ANN models has led to new theoretical results and advances in VLSI technology that make it possible to fabricate microelectronic networks of high complexity. Much of this interest began when ANN's were devised to solve some optimization problems [2]–[4].

The motivation for studying various ANN models for linear algebra problems is twofold. One reason for investigating various ANN models is that basic and fundamental problems such as matrix inversion are often encountered in engineering applications where a very fast on-line solution is required. A second reason is that understanding properties and features of relatively simple ANN models can be an aid to understanding and developing new architectures for more complex and general nonlinear programming problems [3], [31], [32].

Manuscript received July 12, 1990. This paper was recommended by Associate Editor A. Kuh.

A. Cichocki is with the Warsaw Technical University, 00-661 Warsaw, Koszykova 75, Poland.

R. Unbehauen is with the Universitat Erlangen-Nurnberg, D-8520, Erlangen, Germany.

IEEE Log Number 9105049.

II. Formulation of the Basic Problem

Consider the linear parameter estimation model

$$Ax = b + r = b_{\text{true}} \tag{1}$$

where $A = [a_{ij}] \in \mathbb{R}^{m \times n}$ is the matrix model, $b \in \mathbb{R}^m$ is the vector of observations or measurements, $r \in \mathbb{R}^m$ is the unknown vector of measurement errors, $b_{\text{true}} \in \mathbb{R}^m$ is the vector of the true values (usually unknown), and $x = [x_1, x_2, \cdots, x_n]^T \in \mathbb{R}^n$ is the unknown vector of the system parameters to be estimated [17]–[21]. We assume that the number m of equations is greater than or equal to the number n of estimated parameters. It is desired to find in real-time a solution x^* if an exact (error-free) solution exists at all (e.g., for $r = 0$ and a nonsingular matrix A, $x^* = A^{-1}b$) or to find an approximate solution that comes as close as possible to a true solution (i.e., the best estimates of the solution vector x^*) subject to a suitable optimality criterion if the observation vector b is measured with unknown errors.

To formulate the above problem in terms of artificial neural networks (ANN's), the key step is to construct an appropriate computational energy function (Lyapunov function) $E(x)$ so that the lowest energy state will correspond to the desired solution x^*. The derivation of the energy function enables us to transform the minimization problem into a set of ordinary differential or difference equations on the basis of which we will design ANN architectures with appropriate synaptic weights (connection strengths), input excitations, and nonlinear activation functions.

In general, the optimization problem associated with (1) can be formulated as follows.

Find the vector $x^* \in \mathbb{R}^n$ that minimizes the energy function

$$E(x) = \sum_{i=1}^{m} \sigma_i(r_i(x)) \tag{2}$$

where

$$r_i(x) = a_i^T x - b_i = \sum_{j=1}^{n} a_{ij}x_j - b_i \tag{3}$$

1057-7122/92$03.00

represents the residual components of the residend vector

$$r(x) = [r_1(x), r_2(x), \cdots, r_m(x)]^T = Ax - b \quad (4)$$

for a given vector $x(t)$ (actual values) and $\sigma_i(r_i)$ represents suitably chosen convex functions [18]–[24]. In practice, the following cases have special importance.

i) Taking $\sigma_i(r_i) = r_i^2/2$ gives the ordinary least square problem in which we minimize the energy function

$$E_2(x) = \frac{1}{2}\sum_{i=1}^{m} r_i^2(x) = \frac{1}{2}r^T(x)r(x)$$
$$= \frac{1}{2}(Ax - b)^T(Ax - b)$$
$$= \frac{1}{2}\|Ax - b\|_2^2. \quad (5)$$

ii) For $\sigma_i(r_i) = k_i r_i^2/2$ with $k_i > 0$, we have the standard weighted problem with the energy function

$$E_2(x, K) = \frac{1}{2}\sum_{i=1}^{m} k_i r_i^2(x)$$
$$= \frac{1}{2}(Ax - b)^T K(Ax - b) \quad (6)$$
$$\text{with} \quad K = \operatorname{diag}(k_1, k_2, \cdots, k_m).$$

iii) For the logistic function [19]–[21]

$$\sigma_i(r_i) = \frac{\beta}{\alpha}\ln(\cosh(\alpha r_i)) \quad (7a)$$

we have the iteratively reweighted least squares problem[1] with the energy function

$$(x) = E(x, \alpha, \beta)$$
$$= \frac{\beta}{\alpha}\sum_{i=1}^{m}\ln(\cosh(\alpha r_i(x)) \quad (7b)$$

where $\alpha > 0$, $\beta > 0$ are problem-dependent parameters.

iv) For $\sigma_i(r_i) = |r_i|$ we have the least absolute value problem with the energy function

$$E_1(x) = \sum_{i=1}^{m} |r_i(x)|. \quad (8)$$

Apart from the above listed criteria, the l_∞ (or Chebyshev) norm criterion is often used, which can be formulated as the minimax problem [23], [33], [34]

$$\min_{x \in R^n} \max_{1 \le i \le m} \{|r_i(x)|\}. \quad (9)$$

The proper choice of the criterion used depends on the specific applications and greatly on the distribution of the errors in the measurement vector b [18], [19]. The standard least squares criterion is optimal for a Gaussian distribution of the noise [18], [24]. However, the assumption that the set of measurements or observations has a Gaussian error distribution is frequently unrealistic due to different sources of errors such as instrument errors, modeling errors, sampling errors, and human errors. Often some of the measurements contain large errors (called outliers) or wild (spiky) noise. In order to reduce the influence of the outliers, the more robust iteratively reweighted least squares technique (cf., (7)) can be used [19]–[21]. In the presence of outliers (and/or wild noise) an alternative approach is to use the least absolute value (l_1-norm) criterion (cf., (8)) [22]. On the other hand, maximum likelihood considerations suggest the l_∞ or Chebyshev norm criterion if the errors are uniformly distributed [23].

In this paper we will discuss various algorithms and associated circuit architectures for obtaining estimates of the solution vector x^* by using different modifications of the least squares criterion. Algorithms and associated network architectures based on the least absolute value and minimax criteria will be discussed in forthcoming contributions [33], [34].

III. Neuron-Like Architectures for Solving Systems of Linear Equations

There are many different ways to connect neuron-like computing units (cells) into large networks. These different patterns of connections between the cells are called architectures or circuit structures. The purpose of this section is to review known structures [2], [3], [6], [7], [10]–[13] and to propose some new configurations with improved performance and/or with a reduced set of computing units.

3.1. Standard Least Squares Criterion

Using a general gradient approach for minimization of a function the problem formulated by (5) can be mapped to a set of differential equations (an initial value problem) written in the matrix form

$$\frac{dx}{dt} = -\mu(t)\nabla E_2(x)$$
$$\nabla E_2(x) = A^T(Ax - b), \; x(0) = x^{(0)} \quad (10a)$$

where

$$x = [x_1, x_2, \cdots, x_n]^T$$

and $\mu(t) = [\mu_{ij}(t)]$ is an $n \times n$ positive-definite matrix that is often diagonal (i.e., $\mu_{ij} = \mu_j\delta_{ij}$ with $\mu_j > 0$ where δ_{ij} is the Kronecker delta symbol which is 1 if $i = j$ and 0 otherwise), and where $\nabla E_2(x)$ is the gradient of the energy function $E_2(x)$ that can be expressed as

$$\nabla E_2(x) = \left[\frac{\partial E_2(x)}{\partial x_1}, \frac{\partial E_2(x)}{\partial x_2}, \cdots, \frac{\partial E_2(x)}{\partial x_n}\right]^T$$
$$= A^T(Ax - b). \quad (10b)$$

Generally, the entries of the matrix $\mu(t)$ depend on the time and the vector x. The above system of equations can be

[1] The iteratively reweighted least squares method is well known in numerical algebra, and robust statistics are based on repeatedly solving a weighted least squares problem [18]–[21].

written in the following scalar form:

$$\frac{dx_j}{dt} = -\sum_{p=1}^{n} \mu_{jp}\left(\sum_{i=1}^{m} a_{ip}\left(\sum_{k=1}^{n} a_{ik}x_k - b_i\right)\right) \quad (11)$$

with $x_j(0) = x_j^{(0)}$, $m \geq n$, for $j = 1, 2, \cdots, n$.

The basic idea is to compute a trajectory $\boldsymbol{x}(t)$ starting at the initial point $\boldsymbol{x}^{(0)}$ that has the solution $\boldsymbol{x}^*$ as a limit point (for $t \to \infty$).[2] The specific choice of the coefficients $\mu_{jp}(t)$ must ensure the stability of the differential equations and an appropriate convergence speed to the stationary solution (equilibrium) state. It is easy to prove that the system of differential equations (10a) and (b), and (11) is stable (i.e., it has always a stable asymptotic solution) since

$$\frac{dE_2}{dt} = \sum_{j=1}^{n} \frac{\partial E_2(\boldsymbol{x})}{\partial x_j}\frac{dx_j}{dt} = \left(\nabla E_2(\boldsymbol{x})\right)^T \frac{d\boldsymbol{x}}{dt}$$

$$= -\left(\nabla E_2(\boldsymbol{x})\right)^T \boldsymbol{\mu}(t) \nabla E_2(\boldsymbol{x}) \leqq 0 \quad (12)$$

under the condition that the matrix $\boldsymbol{\mu}(t)$ is positive-definite for all values of $\boldsymbol{x}$ and t, and in absence of round-off errors in the matrix $\boldsymbol{A}$.

3.2. Iteratively Reweighted Least Squares Criterion

Although the standard (ordinary) least squares criterion is optimal for a Gaussian error distribution, it can, nevertheless, provide very poor estimates of the vector $\boldsymbol{x}^*$ in the presence of outliers or wild (spiky) noise in the observation (measurement) vector $\boldsymbol{b}$. In order to diminish (reduce) the influence of the outliers (i.e., to provide a more robust estimate of the vector $\boldsymbol{x}^*$) we will employ the iteratively reweighted least squares criterion [18]–[21].

Applying the gradient approach for the minimization of the energy function $E(\boldsymbol{x}, \alpha, \beta)$ given by (7b) we obtain the system of differential equations

$$\frac{dx_j}{dt} = -\sum_{p=1}^{n} \mu_{jp}\left(\sum_{i=1}^{m}\left(a_{ip} g_i\left[\sum_{k=1}^{n} a_{ik}x_k - b_i\right]\right)\right),$$

$$j = 1, 2, \cdots, n \quad \text{with} \, x_j(0) = x_j^{(0)} \quad (13)$$

where

$$g_i[r_i] = g_i\left[\sum_{k=1}^{n} a_{ik}x_k - b_i\right]$$

is a sigmoid activation function developed as

$$g_i[r_i] = \frac{\partial \sigma_L(r_i)}{\partial r_i} = \frac{\partial\left[\dfrac{\beta}{\alpha}\ln\left(\cosh\left(\alpha r_i\right)\right)\right]}{\partial r_i} = \beta \tanh\left(\alpha r_i\right). \quad (14)$$

[2] It should be noted that we are concerned only with finding the limit (equilibrium or stationary point) as the time tends to infinity rather than determining a detailed and accurate picture of the whole trajectory itself.

The above system of differential equations can be rewritten as the set of nonlinear equations

$$e_i(\boldsymbol{x}) = g_i\left[\sum_{k=1}^{n} a_{ik}x_k - b_i\right], \qquad i = 1, 2, \cdots, m \quad (15a)$$

$$\frac{\partial \epsilon(\boldsymbol{x})}{\partial x_p} = \sum_{i=1}^{m} a_{ip} e_i(\boldsymbol{x}), \; p = 1, 2, \cdots, n \quad (15b)$$

$$\frac{dx_j}{dt} = -\sum_{p=1}^{n} \mu_{jp}\frac{\partial \epsilon(\boldsymbol{x})}{\partial x_p}, \; x_j(0) = x_j^{(0)}, \; j = 1, 2, \cdots, n \quad (15c)$$

which can be implemented directly by a new artificial neural network depicted in Fig. 1. The architecture of the network resembles the general Tank–Hopfield model [2], [13]. However, the network employs sigmoid nonlinearities not to provide binary (quantized) outputs, but in order to improve the quality of the continuous-valued solution in the presence of outliers or spiky noise. The circuit consists of (continuous-time or possibly discrete-time) integrators and adders (summing amplifiers) with associated connection weights. The nonlinearities $g_i(r_i)$ are not shown explicitly in Fig. 1, and it is assumed that these nonlinearities are incorporated only in the first (input) layer of artificial neurons. The connection weights (synaptic strengths) denoted by a_{ij}, $\tilde{a}_{ij}$, and μ_{ij} represent the coefficients of the matrices $\boldsymbol{A}$ and $\boldsymbol{\mu}$. These weights can be fixed or time-variable depending on the entries of the matrices $\boldsymbol{A}$ and $\boldsymbol{\mu}$.

In practice they can be implemented by using VLSI analog multipliers or programmable (voltage controlled) resistors [1]–[3]. The time-variable connection weights $a_{ij}(t)$ can be implemented by CMOS analog multipliers or multiplying digital-to-analog converters [1].

The architecture shown in Fig. 1 also somewhat resembles an earlier version of the perceptron [9]. The circuit consists of three layers of artificial neurons (which are connected in a feedforward mode). Each neuron produces its output by computing the inner product of its input signals and its appropriate weight vector and by possibly passing the result through the nonlinear sigmoid function $g_i(r_i)$.

The first layer is sometimes called the "sensors layer" (since it senses the actual variables $x_j(t)$ and computes the actual errors $e_i(\boldsymbol{x})$ defined by (15a)). This computation should be made as accurately as possible (although the shape of the nonlinearities is not essential). The signals $e_i(\boldsymbol{x})$ produced by the "sensors layer" are combined in the second layer of computing cells called "association elements" that compute or rather only estimate the gradient components $\partial \epsilon(\boldsymbol{x})/\partial x_p$ (cf., (15b)). Since for a well-conditioned system of equations only a crude estimation of the gradient components is needed, the connection weights denoted by $\tilde{a}_{ij}$ can often be realized approximately, i.e., $\tilde{a}_{ij} \cong a_{ij}$. As shown later for some classes of matrices the circuit exhibits notable robustness with respect to variations of the weights $\tilde{a}_{ij}$ and μ_{ij}. The third layer comprises "response elements" and constitutes the proper learning system, i.e., the gains μ_{ij} are usually

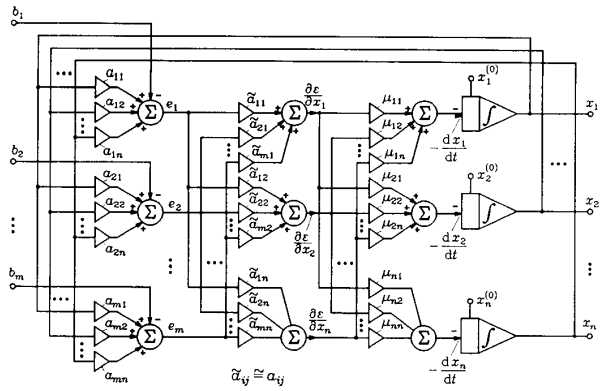


Fig. 1. Schematic architecture of an artificial neural network for solving a system of linear equations $Ax \approx b$. The optional nonlinear activation functions used in the first layer of the computing units are not shown explicitly for simplicity (cf., (15), (18), and (19)).

made variable during the minimization process.[3] By learning we understand here the process that improves the properties (increased convergence rate, decreased settling time) and the functionality of the network in the course of time. The selection of appropriate weights μ_{ij} is a crucial problem. Small values of μ_{ij} (which correspond to large time constants of the integrators) may make the convergence speed to the equilibrium (stationary) state (i.e., the desired solution) very slow. On the other hand, a large μ_{ij} may lead to an unstable behavior of the network (especially if discrete-time switched-capacitor integrators are used). In the simplest case we can choose the weights μ_{ij} constant. However, it should be emphasized that an adaptive selection of the $\mu_{ij}(t)$ can greatly increase the convergence rate without causing stability problems as could arise by use of higher (fixed) constant values of μ_{ij} (cf., Section 4.3).

The use of sigmoid nonlinearities in the first layer of "neurons" is essential for overdetermined linear systems of equations since it enables us to obtain more robust solutions, which are less insensitive to outliers (in comparison to the standard linear implementation given by (11)), by compressing large residuals and preventing their absolute values from being greater than the prescribed cut-off parameter $\beta > 0$. Note that if the cut-off parameter β is chosen large with $\alpha = 1/\beta$, then we will come close to the standard least squares criterion (cf., (11)–(14)). On the other hand, taking $\beta = 1$ and α large gives something similar to the l_1-norm (least absolute value) criterion, where the sum of the absolute values of the residuals is minimized since then the activation function described by (14) closely approximates the signum (hard limiter) function (cf., Example 4).

In order to reduce the influence of the outliers many different weighting functions $\sigma_i(r_i)$ have been proposed in robust statistics [19]–[21]. A detailed discussion of the application of the various possible weighting functions in ANN implementations is out of the scope of this paper. We will give here only two typical examples. For example, instead of the logistic function defined by (7a) we can employ Huber's function determined as [18]–[21]

$$\sigma_H(r_i) = \begin{cases} r_i^2/2, & \text{for } |r_i| \le \beta \\ \beta|r_i| - \beta^2/2, & \text{for } |r_i| > \beta \end{cases} \tag{16}$$

or Huber's function with saturation also called Talvar's function [21]

$$\sigma_T(r_i) = \begin{cases} r_i^2/2, & \text{for } |r_i| \le \beta \\ \beta^2/2, & \text{for } |r_i| > \beta. \end{cases} \tag{17}$$

For such weighting functions $\sigma(r_i)$ the nonlinear activation function used in (13) and (15a) takes the form:

$$e_i(x) = g_i(r_i(x)) = \frac{\partial \sigma_H(r_i(x))}{\partial r_i} = \begin{cases} -\beta, & \text{for } r_i < -\beta \\ r_i(x), & \text{for } |r_i| \le \beta \\ \beta, & \text{for } r_i > \beta \end{cases} \tag{18}$$

for Huber's function, and

$$e_i(x) = g_i(r_i(x)) = \frac{\partial \sigma_T(r_i(x))}{\partial r_i} = \begin{cases} r_i(x), & \text{for } |r_i| \le \beta \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

for Talvar's function.

The activation function defined by (19) provides that all equations (in the set of equations (1)) with large absolute values of the residuals above the cut-off parameter β (i.e., $|r_i(x^*)| > \beta$) are neglected (i.e., the corresponding neurons in the network shown in Fig. 1 are inhibited) and they have no influence on the final solution.

It is interesting to note that the activation functions $e_i(x) = g_i(r_i(x))$ given by (18) have a direct implementation in CMOS technology, since such activation functions are naturally provided by the saturating characteristic of summing amplifiers (adders); i.e., if we take the power supply voltages as $\pm\beta$ then the output voltage of the corresponding amplifiers can linearly vary in the range from $-\beta$ to β. However, it is important to note that due to the saturation characteristic of the amplifiers used, in practical implementations all signals in the network shown in Fig. 1 are in fact bounded between the limits determined by the power supply voltages. If the variation of the output signals $x_j(t)$ is large, above the power supply voltages, then the saturation effect may introduce more than one attractor for the network quite different from the desired one described by (13).[4] To avoid this problem the system of linear equations should be appropriately scaled and/or two different power supply voltages $\pm\beta$ and $\pm V_{cc}$ with $V_{cc} \gg \beta$ can be used. There are two alternative VLSI implementations of ANN's. The first approach uses continuous-time computing units that simulate appropri-

[3] For this reason we will call the matrix μ the learning matrix.

[4] The authors are grateful to one of the referees for pointing out this problem.

ate differential equations (cf., (13)). In the second approach the differential equations are mapped into corresponding (generally nonlinear) difference equations that are simulated by discrete-time computing units (i.e., discrete-time integrators and summing amplifiers) [1]. For example, by applying the standard first-order discretization technique (Euler integration rule) to the system of differential equations (13) (with $\mu(t) = \text{diag}(\mu_1, \mu_2, \cdots, \mu_n)$), this system can be converted to the difference equations

$$x_j^{(k+1)} = x_j^{(k)} - \alpha_j^{(k)} \sum_{i=1}^{m} a_{ij} g_i \left[\sum_{j=1}^{n} a_{ij} x_j^{(k)} - b_i \right] \quad (20)$$

with $x_j^{(0)} = x_j(0)$

for $j = 1, 2, \cdots, n; \quad k = 0, 1, 2, \cdots$

where $x_j^{(k)} = x_j(k\tau)$ with the integration step τ and $\alpha_j^{(k)} = \tau \mu_j(k\tau) > 0$. (It has been assumed for simplicity that the matrix $\boldsymbol{\mu}(t)$ is the diagonal matrix with the coefficients $\mu_j(t) > 0$); $g_i(\cdot)$ is the nonlinear activation function given by (14), (18), or (19).)

The above system of difference equations can easily be implemented by using switched-capacitor (SC) techniques [1]. It should be noted that for the discrete-time (iterative) algorithm described by (20) the controlling parameters $\alpha_j^{(k)}$ must be bounded in a small range, i.e., $0 < \alpha_j^{(k)} \leqq \alpha_{\max}$, to assure stability of the algorithms. However, a small parameter α means that the convergence to a solution is slow, while a large α means that oscillations may occur and stability may be lost. On the other hand, for the continuous-time algorithm given by (13) the parameters $\mu_j > 0$ can be set to a theoretically arbitrarily large value without affecting the stability of the system.

3.3. Special Cases with Simpler Architectures

For simplicity, in our further considerations, we will neglect the nonlinearities $g_i(\cdot)$ of the computing units unless they are explicitly introduced.

The circuit structure shown in Fig. 1 is highly redundant, i.e., it can be considerably simplified for some important special cases. The simplified ANN architectures have the advantage of relatively few interconnections which is an attractive feature from the VLSI fabrication point of view.

An important class of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} (\boldsymbol{A} \in \Re^{n \times n})$ problems are the well-scaled and well-conditioned problems for which the condition number [16] is not large, i.e., the eigenvalues of the matrix $\boldsymbol{A}$ are clustered in a set containing eigenvalues of similar magnitude. In such a case the matrix differential equation (10) can be modified to the form

$$\frac{d\boldsymbol{x}}{dt} = -\mu_0 \tilde{\boldsymbol{A}}^T (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}), \quad \boldsymbol{x}(0) = \boldsymbol{x}^{(0)} \quad (21)$$

where $\mu_0(t)$ is a positive scalar coefficient and $\tilde{\boldsymbol{A}}$ is an arbitrary $n \times n$ nonsingular matrix that should be chosen such that the matrix $\boldsymbol{C} = \tilde{\boldsymbol{A}}^T \boldsymbol{A}$ is a positive stable (Hurwitz) matrix (i.e., all the eigenvalues of the matrix $\boldsymbol{C} = \tilde{\boldsymbol{A}}^T \boldsymbol{A}$ must occur in the right half of the complex plane to assure stability).[5] It should be noted that the stable equilibrium point $\boldsymbol{x}^*$ (for $d\boldsymbol{x}/dt = \boldsymbol{0}$) does not depend on the value μ_0 and the coefficients of the matrix $\tilde{\boldsymbol{A}}$. As a special case the matrix $\tilde{\boldsymbol{A}}$ can be a diagonal matrix with entries ± 1, i.e., the set of differential equations can take the form

$$\frac{dx_j}{dt} = \mp \mu_0 g_j \left(\sum_{k=1}^{n} a_{jk} x_k - b_j \right), \quad x_j(0) = x_j^{(0)},$$

for $j = 1, 2, \cdots, n \quad (22)$

where $g_j(\cdot)$ is the nonlinear sigmoid function defined by (14).

It should be pointed out that the use of sigmoid nonlinearities $g_i(\cdot)$, in the special case that the matrix $\boldsymbol{A}$ is nonsingular, is nonessential and has no influence on the final solution $\boldsymbol{x}^*$, since all residuals $r_i(\boldsymbol{x}(t))$ tend to zero as $\boldsymbol{x}(t) \to \boldsymbol{x}^*$. However, such nonlinearities may have influence on the settling time (convergence speed).

A very important class of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ problems is that where $\boldsymbol{A} \in \Re^{n \times n}$ is a nonsingular, symmetric, and positive-definite matrix (i.e., $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} > 0$ for all nonzero $\boldsymbol{x} \in \Re^n$). For such a case, choosing

$$\boldsymbol{\mu} := \mu_0 \boldsymbol{A}^{-1} \quad \text{with } \mu_0(t) > 0 \quad (23)$$

the matrix differential equation (10) simplifies to the form

$$\frac{d\boldsymbol{x}}{dt} = -\mu_0 (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \quad \text{with } \boldsymbol{x}(0) = \boldsymbol{x}^{(0)}. \quad (24)$$

It is interesting to note that the above problem $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with symmetric, positive-definite matrix $\boldsymbol{A}$ is equivalent to the unconstrained quadratic programming problem

$$\min_{\boldsymbol{x} \in \Re^n} E(\boldsymbol{x}), \quad E(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x} \quad (25)$$

since from the gradient $\nabla E(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$ and the Hessian $\nabla^2 E(\boldsymbol{x}) = \boldsymbol{A}$ it follows that $\boldsymbol{x}^* = \boldsymbol{A}^{-1}\boldsymbol{b}$ is the unique global minimum of the energy function $E(\boldsymbol{x})$. However, if $\boldsymbol{A}$ is positive semi-definite, a stationary point (if it exists) is a weak local minimum of $E(\boldsymbol{x})$. If $\boldsymbol{A}$ is indefinite and nonsingular, $\boldsymbol{x}^*$ is a saddle point, and $E(\boldsymbol{x})$ is unbounded from above and below. A common technique for converting a system of linear equations with an indefinite matrix $\boldsymbol{A}$ to a system with a positive-definite matrix is to transform it to the normal equation

$$\boldsymbol{A}^T \boldsymbol{A} \boldsymbol{x} = \boldsymbol{A}^T \boldsymbol{b}. \quad (26)$$

Unfortunately, for ill-conditioned problems it cannot be recommended to transform $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ to the normal equation since the formation of the matrix $\boldsymbol{A}^T\boldsymbol{A}$ can cause perturbation (underflow and overflow) errors and can square the conditioning (condition number) of the problem in comparison with a method which used $\boldsymbol{A}$ directly [16].

[5] A matrix $\boldsymbol{A}$ is positive (negative) stable if the real parts of all of its eigenvalues are positive (negative). Clearly, if the matrix $\boldsymbol{A}$ is positive stable, then $-\boldsymbol{A}$ is negative stable and vice versa. Of course, a positive definite matrix is positive stable.

For some well-conditioned problems, instead of minimizing one global energy function $E(x)$ (cf. (2)), it is possible to minimize simultaneously n local energy functions defined by

$$E_i(x) = |r_i| = \left| \sum_{k=1}^{n} a_{ik} x_k - b_i \right| \tag{27}$$

or

$$E_i(x) = \frac{1}{2} r_i^2 = \frac{1}{2} \left(\sum_{k=1}^{n} a_{ik} x_k - b_i \right)^2, \quad \text{for } i = 1, 2, \cdots, n. \tag{28}$$

Applying a general gradient method for each energy function given by (27) we have

$$\frac{dx_j}{dt} = -\mu_j a_{jj} \operatorname{sign}(r_j), \quad j = 1, 2, \cdots, n \tag{29a}$$

where

$$\mu_j(t) > 0$$

and

$$\operatorname{sign}(r_j) = \begin{cases} 1, & \text{for } E_j(x) \geqq 0 \\ -1, & \text{otherwise.} \end{cases} \tag{29b}$$

Assuming now that

$$\mu_j(t) = \frac{\mu_0(t)}{|a_{jj}|} > 0, \quad \text{for } j = 1, 2, \cdots, n \tag{30a}$$

we get

$$\frac{dx_j}{dt} = -\mu_0 \operatorname{sign}(a_{jj} r_j). \tag{30b}$$

Analogously, for the energy function given by (28) we can obtain

$$\frac{dx_j}{dt} = -\mu_j r_j \frac{dr_j}{dx_j} = -\mu_j r_j a_{jj}. \tag{31}$$

Assuming, for example, that

$$\mu_j = \frac{\mu_0(t)}{|a_{jj}|} > 0, \quad j = 1, 2, \cdots, n \tag{32a}$$

we have

$$\frac{dx_j}{dt} = -\mu_0 r_j \operatorname{sign}(a_{jj}). \tag{32b}$$

For each set of differential equations (cf., (29 a), (b), (30 a), (b), (31), and (32 a), (b)) we can easily construct appropriate neural networks. An exemplary realization of the set of equations (30 a), (b) is depicted in Fig. 2. The circuit principally consists of adders, voltage comparators, and integrators. Depending on the sign of the coefficient $r_j a_{jj}$, a noninverting or inverting comparator is used in the corresponding channel.

To find sufficient conditions for the stability of such a circuit we can use the Lyapunov method. For example, for

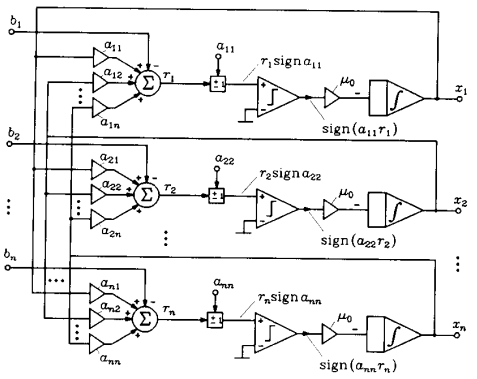


Fig. 2. Exemplary simplified architecture of an ANN for the solution of a system of linear equations with a diagonally dominant matrix (cf., (30)).

(27) and (30) we get

$$\frac{dE_i}{dt} = \sum_{j=1}^{n} \frac{\partial E_i}{\partial x_j} \frac{dx_j}{dt} = -\mu_0 \sum_{j=1}^{n} a_{ij} \operatorname{sign}(r_i) \operatorname{sign}(a_{jj} r_j)$$

$$= -\mu_0 \left[a_{ii} \operatorname{sign}(a_{ii} r_i^2) - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} \operatorname{sign}(a_{jj} r_j r_i) \right]. \tag{33}$$

Hence we estimate that

$$\frac{dE_i(x)}{dt} < 0 \quad \text{for each } i \tag{34a}$$

if

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|. \tag{34b}$$

This means that the system is stable if the matrix A is diagonally dominant.

In a similar way we can derive a sufficient condition for the stability of the system given by (29), (31), and (32) (cf., Table I).

3.4. Positive Connection

In some practical implementations of ANN's it is convenient to have all gains (connection weights) a_{ij} positive. This can easily be achieved by the extension of the system of linear equations

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{m1} & \cdots & & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{35}$$

TABLE I
VARIOUS CONTINUOUS-TIME COMPUTATION MODELS OF ANN'S FOR THE SOLUTION OF A SYSTEM OF LINEAR EQUATIONS

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\,(e_i(\boldsymbol{x}) := g_i\Big(\sum_{k=1}^{n} a_{ik}x_k - b_i\Big),\ j = 1, 2, \cdots, n,\ i = 1, 2, \cdots, m)$$

No.	Differential equation	Sufficient conditions for stability
1	$\frac{dx_j}{dt} = -\mu_j \sum_{i=1}^{m} a_{ij} e_i(\boldsymbol{x})$	matrix $\boldsymbol{A}$ arbitrary $\mu_j(t) > 0$
2	$\frac{dx_j}{dt} = -\mu_j \operatorname{sign}\left[\sum_{i=1}^{m} a_{ij} e_i(\boldsymbol{x})\right]$	matrix $\boldsymbol{A}$ arbitrary $0 < \mu_j(t) < \mu_{\max}$
3	$\frac{dx_j}{dt} = -\mu_j \sum_{i=1}^{n} \tilde{a}_{ij} e_i(\boldsymbol{x})$	matrix $\boldsymbol{C} = \tilde{\boldsymbol{A}}^T\boldsymbol{A}$ must be positive stable $\mu_j(t) > 0$
4	$\frac{dx_j}{dt} = -\mu_j e_j(\boldsymbol{x})$	matrix $\boldsymbol{A} = [a_{ij}]_{n\times n}$ must be positive stable $\mu_j(t) > 0$
5	$\frac{dx_j}{dt} = -\mu_0 e_j(\boldsymbol{x}) \operatorname{sign}[a_{jj}]$	$\det \boldsymbol{A} \neq 0, \quad \mu_0 > 0$ $\lvert a_{ii}\rvert > \sum_{\substack{j=1 \\ j\neq i}}^{n} \lvert a_{ij}\rvert$ for $i = 1, \cdots, n$
6	$\frac{dx_j}{dt} = -\mu_0 a_{jj} \operatorname{sign}[e_j(\boldsymbol{x})]$	$\det \boldsymbol{A} \neq 0, \quad 0 < \mu_0 < \mu_{\max}$ $a_{ii} > \sum_{\substack{j=1 \\ j\neq i}}^{n} \lvert a_{ij}a_{jj}\rvert$
7	$\frac{dx_j}{dt} = -\mu_0 \operatorname{sign}[a_{jj}e_j(\boldsymbol{x})]$	$\det \boldsymbol{A} \neq 0 \quad 0 < \mu_0 < \mu_{\max}$ $\lvert a_{ii}\rvert \sum_{\substack{j=1 \\ j\neq i}}^{n} \lvert a_{ij}\rvert$ for $i = 1, \cdots, n$
8	$\frac{dx_j}{dt} = -\mu_j \sum_{i=1}^{m} a_{ij}(k_i e_i(\boldsymbol{x}) + \lambda_i)$ $\frac{d\lambda_i}{dt} = \rho_i(e_i(\boldsymbol{x}) - \alpha\lambda_i)$	matrix $\boldsymbol{A}$ arbitrary $\mu_j(t) > 0, \quad \rho_i(t) \geqq 0$ $0 \leqq \alpha \leqq 1, \quad k_i \geqq 0$
9	$\frac{dx_j}{dt} = -\mu_j(k_j e_j(\boldsymbol{x}) + \lambda_j)$ $\frac{d\lambda_j}{dt} = \rho_j(e_j(\boldsymbol{x}) - \alpha\lambda_j)$	matrix $\boldsymbol{A}$ must be positive stable $\mu_j(t) > 0, \quad \rho_j(t) \geqq 0$ $0 \leqq \alpha \leqq 1, \quad k_j \geqq 0$
10	$\frac{dx_j}{dt} = -\mu_j \operatorname{sign}[k_j e_j(\boldsymbol{x}) + \lambda_j]$ $\frac{d\lambda_j}{dt} = \rho_j(e_j(\boldsymbol{x}) - \alpha\lambda_j)$	matrix $\boldsymbol{A}$ must be positive stable $0 < \mu_j(t) \leqq \mu_{\max}, \quad \rho_j(t) \geqq 0$ $0 \leqq \alpha \leqq 1, \quad k_j \geqq 0$

with different signs of the entries a_{ij} to the form

$$\begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} & \cdots & \hat{a}_{1n} & \hat{a}_{1,n+1} \\ \vdots & & & & \\ \hat{a}_{m1} & \hat{a}_{m2} & \cdots & \hat{a}_{mn} & \hat{a}_{m,n+1} \\ \hat{a}_{m+1,1} & \hat{a}_{m+1,2} & \cdots & \hat{a}_{m+1,n} & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \\ 0 \end{bmatrix} \quad (36)$$

with all entries $\hat{a}_{ij}$ positive. Since both systems of linear equations (35) and (36) must be equivalent with respect to the variables $x_1, x_2, \cdots, x_n$ the following relation must be satisfied:

$$\boldsymbol{A} + \left[\hat{a}_{1,n+1}, \hat{a}_{2,n+1}, \cdots, \hat{a}_{m,n+1}\right]^T \cdot \left[\hat{a}_{m+1,1}, \hat{a}_{m+1,2}, \cdots, \hat{a}_{m+1,n}\right] = \hat{\boldsymbol{A}} \quad (37)$$

where $\boldsymbol{A} = [a_{ij}]_{m\times n}$ and $\hat{\boldsymbol{A}} = [\hat{a}_{ij}]_{m\times n}$ are $m \times n$ matrices described above. From (37) we obtain

$$\hat{a}_{ij} = a_{ij} + \hat{a}_{i,n+1}\hat{a}_{m+1,j} \geqq 0. \quad (38)$$

From the above formula it is evident that we can always choose auxiliary entries $\hat{a}_{m+1,j}$ and $\hat{a}_{i,n+1}$ ($i = 1, 2, \cdots, m$; $j = 1, 2, \cdots, n$) so that all entries $\hat{a}_{ij}$ will be positive. Thus, instead of solving the original problem (35), it is possible to solve the problem (36) with all the entries (connection weights) a_{ij} positive.

IV. Improved Circuit Structures for Ill-Conditioned Problems

The above proposed schemes and structures are suitable for well-conditioned problems. However, for ill-conditioned problems [16], [25]–[27] such schemes may be prohibitively slow and they may even fail to find an appropriate solution or they may find a solution with large error. This can be explained by the fact that for an ill-conditioned problem we may obtain a system of stiff differential equations [15]. Loosely speaking the system of stiff differential equations is one that is stable but exhibits a wide difference in the behavior of the individual components of the solution. The essence of a system of stiff differential equations is that one has a very slowly varying solution (trajectory) which is such that some perturbation to it is rapidly damped [15]. For a linear system of differential equations $\dot{x} = -\mu_0(Ax - b)$ this happens when the time constants of the system, i.e., the reciprocals of the eigenvalues of the matrix A are widely different.

4.1. Augmented Lagrangian with Regularization

Motivated by the desire to alleviate the stiffness of the differential equations and simultaneously to improve the convergence properties and the accuracy of the desired networks we will develop a new ANN architecture with improved performance. For this purpose we construct the following energy function (augmented Lagrangian function) for the problem (1):

$$E(x) = \frac{1}{2} r^T(x) K r(x) + \lambda^T r(x) - \frac{\alpha}{2}\lambda^T\lambda = \frac{1}{2}\sum_{i=1}^{m} k_i r_i^2(x) + \sum_{i=1}^{m}\left(\lambda_i r_i(x) - \frac{\alpha}{2}\lambda_i^2\right) \quad (39)$$

where

$$r(x) = Ax - b = [r_1, r_2, \cdots, r_m]^T$$

i.e.,

$r_i(x) = \sum_{k=1}^{n} a_{ik}x_k - b_i$—the residuals,
$\lambda = [\lambda_1, \lambda_2, \cdots, \lambda_m]^T$—the Lagrange multipliers,
$K = \text{diag}(k_1, k_2, \cdots, k_m)$—the weighting penalty coefficients ($0 \leqq k_i \leqq 1$),
$\alpha \geqq 0$—the regularization parameter.

The augmented Lagrangian is obtained from the ordinary (common) Lagrangian by adding penalty terms [28], [29], [31], [32]. Since an augmented Lagrangian can be ill-conditioned a regularization term with coefficient α is introduced to eliminate the instabilities associated with the penalty terms. The problem of minimization of the above defined energy function can be transferred to the set of differential equations

$$\frac{dx_j}{dt} = -\mu_j\sum_{i=1}^{m}(k_i r_i(x) + \lambda_i)a_{ij} \quad (40a)$$

$$\frac{d\lambda_i}{dt} = \rho_i(r_i(x) - \alpha\lambda_i) \quad (40b)$$

for $j = 1, 2, \cdots, n$; $i = 1, 2, \cdots, m$ and with $\mu_j > 0$ and $\rho_i > 0$.

The above set of equations can be written in the compact matrix form

$$\frac{dx}{dt} = -\mu A^T y \quad (41a)$$

$$\frac{d\lambda}{dt} = \rho[(Ax - b) - \alpha\lambda] \quad (41b)$$

where

$y = [\lambda_1 + k_1 r_1, \lambda_2 + k_2 r_2, \cdots, \lambda_m + k_m r_m]^T$,
$x = [x_1, x_2, \cdots, x_n]^T$,
$\lambda = [\lambda_1, \lambda_2, \cdots, \lambda_m]^T$,
$\mu = \text{diag}(\mu_1, \mu_2, \cdots, \mu_n)$, $\mu_j > 0$,
$\rho = \text{diag}(\rho_1, \rho_2, \cdots, \rho_m)$, $\rho_i > 0$,
$\alpha \geqq 0$.

On the basis of this set of differential equations we can design the general circuit structure depicted in Fig. 3. In comparison to the architecture given in Fig. 1, the circuit contains extra damped integrators and amplifiers (gains k_i). The addition of these extra gains and integrators does not change the stationary point x^* but, as shown by computer simulation experiments, helps to damp parasitic oscillations, improves the final accuracy, and increases the convergence speed (decreases the settling time) (cf., Section VI).

Depending on the properties of the matrix A various simplified configurations of the general architecture are possible (cf., Table I). Analogous to our previous considerations auxiliary sigmoid nonlinearities can be incorporated in the first layer of computing units (i.e., adders) in order to reduce the influence of outliers. The performance and computer simulations of the proposed architectures are discussed later in this paper.

4.2. Preconditioning

Preconditioning techniques [16], [35] form a class of linear transformations of the matrix A or the vector x that improve the eigenvalue structure of the Hessian of the specified energy function and alleviate the stiffness of the associated system of differential equations. The simplest technique that enables us to incorporate preconditioning in an ANN implementation is to apply a linear transformation $x = My$ where M is an appropriate matrix, i.e., instead of minimizing the energy function (5) we can minimize the modified energy function

$$E_2(y) = \frac{1}{2}\|AMy - b\|_2^2. \quad (42)$$

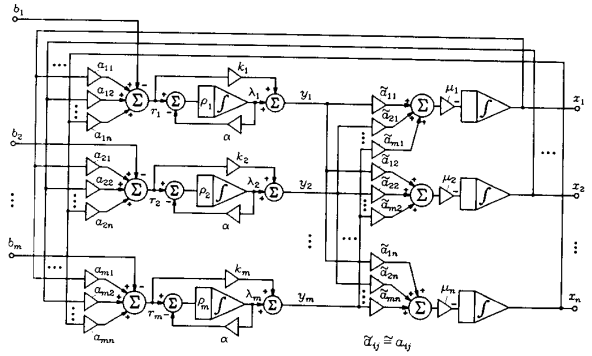


Fig. 3. General architecture of an ANN for matrix inversion (cf., (40)).

The above problem can be solved by the simulating system of differential equations

$$\frac{dy}{dt} = -\mu_0(AM)^T(AMy - b),\ y(0) = y^{(0)} \quad (43)$$

where μ_0 is a positive scalar coefficient. Multiplying (43) by the matrix M, we get

$$\frac{dx}{dt} = -(\mu_0 MM^T)A^T(Ax - b),\ x(0) = x^{(0)}. \quad (44)$$

Setting $\mu = \mu_0 MM^T$ we get a system of differential equations already considered in Section 3.1. Thus the realization of a suitable symmetric positive-definite matrix μ instead of a simple scalar $\mu_0 > 0$ enables us to perform preconditioning, which may considerably improve the convergence properties of the system [16], [35].

4.3. Artificial Neural Network with Time Processing Independent of the Size of the Problem

The systems of differential equations considered above cause the trajectory $x(t)$ to converge to a desired solution x^* only for $t \to \infty$, although the convergence speed can be very high. In some real-time applications it is required to assure that the specified energy function $E(x)$ reach the minimum at a prescribed finite period of time, say t_r, (i.e., $x(t_r) = x^*$) or that $E(x)$ becomes close to the minimum with a specified error $\delta > 0$ (where δ is an arbitrarily chosen positive very small number). In other words, we can define the reachability time t_r as the settling time after which the energy function $E(x)$ enters a δ-neighborhood of the minimum and remains there ever after the moment t_r. Such a problem can be solved by making the coefficients $\mu_{ij}(t)$ of the matrix $\mu(t)$ (cf., (10)) adaptive during the minimization process, under the assumption that the initial value $E(x^{(0)})$ and the minimum (final) value $E(x^*)$ of the energy function $E(x(t))$ are known or can be estimated.

Let us consider the $Ax = b$ problem with a nonsingular matrix A, which can be mapped to the system of differential equations (cf., (10) and (21))

$$\frac{dx}{dt} = -\mu_0(t)A^T(Ax - b) \quad \text{with } x^{(0)} = 0 \quad (45a)$$

where the adaptive parameter $\mu_0(t)$ can be defined as

$$\mu_0(t) = \begin{cases} \mu_{0\max} & \text{for } E(x(t)) = 0, \\ \dfrac{\mu}{r^T AA^T r} & \text{otherwise,} \end{cases} \quad (45b)$$

$$r = Ax - b,\ \mu = \frac{1}{\tau} > 0\ (\tau \text{ integration time constant}).$$

Note that for this problem $E(x^{(0)}) = (1/2)\sum_{i=1}^{n} b_i^2$ and $E(x^*) = 0$.

Consider the time derivative of the energy function given by (5) associated with the system of differential equations (45) as

$$\frac{dE}{dt} = \sum_{j=1}^{n} \frac{\partial E}{\partial x_j}\frac{dx_j}{dt} = \left[\nabla E(x)\right]^T \frac{dx}{dt}$$

$$= -\mu_0(t)(Ax - b)^T AA^T(Ax - b)$$

$$= -\mu_0(t) r^T AA^T r = -\mu < 0 \quad \text{for } \frac{dx}{dt} \neq 0 \quad (46)$$

and $dE/dt = 0$ only for $dx/dt = 0$.

Hence it follows that the energy function decreases in time linearly during the minimization process as

$$E(t) = E(x^{(0)}) - \mu t \quad (47)$$

and reaches the value $\delta = 0$ (very close to the minimum) after the time

$$t_r = \frac{E(x^{(0)}) - \delta}{\mu} = \frac{\frac{1}{2}\sum_{i=0}^{n} b_i^2 - \delta}{\mu}. \quad (48)$$

By choosing $\mu = E(x^{(0)})/t_{\max}$ we find that the system of equations (45) reaches the equilibrium (stationary point) in the prescribed time $t_r \cong t_{\max}$ independent of the size of the problem. The system of differential equations (45) can (approximately) be implemented by the ANN shown in Fig. 4 employing auxiliary analog multipliers and dividers[6] (cf., Example 4). The main disadvantage of the proposed architecture is the requirement to use extra, relatively expensive analog multipliers and dividers [1].

V. Neural Networks for Linear Programming

The ANN architectures considered in the previous sections can easily be employed for the solution of a linear programming problem [2], [3], [6], [7], [13], [14] which can be stated in standard form as follows.

Minimize the scalar cost function

$$F(x) = \sum_{j=1}^{n} c_j x_j \quad (49a)$$

[6] For a very small value of the signal $y_0 := \sum_{i=1}^{m} y_i^2 = r^T AA^T r$ the dividers may saturate. To prevent this a limiter is used that assures that the divisor signal $v(t)$ is greater than or equal to a small positive constant ϵ. Alternatively, a small constant ϵ can be added to the signal y_0.

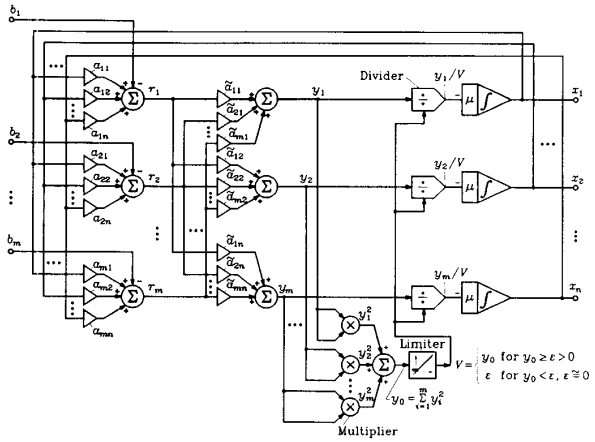


Fig. 4. Architecture of an ANN providing linearly decreasing the energy function in time in a prescribed speed of convergence (cf., (45)).

subject to the linear constraints

$$r_i(\boldsymbol{x}) = \sum_{j=1}^{n} a_{ij}x_j - b_i = 0,\ x_j \geqq 0 \qquad (49b)$$

for $i = 1, 2, \cdots, m$.

By use of the modified Lagrange multiplier approach we can construct the computation energy function

$$E(\boldsymbol{x}) = \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} \lambda_i r_i - \frac{1}{2}\alpha \sum_{i=1}^{m} \lambda_i^2 \qquad (50)$$

where α is the regularization parameter. The problem of minimization of the energy function $E(\boldsymbol{x})$ can be transformed to a set of differential equations

$$\frac{dx_j}{dt} = -\mu_j\left(c_j + \sum_{i=1}^{m} a_{ij}\lambda_i\right),\ x_j(0) = x_j^{(0)},\ x_j \geqq 0 \qquad (51a)$$

$$\frac{d\lambda_i}{dt} = \rho_i\left(\sum_{j=1}^{n} a_{ij}x_j - b_i - \alpha\lambda_i\right),\ \lambda_i(0) = \lambda_i^{(0)} \qquad (51b)$$

for $j = 1, 2, \cdots, n$; $i = 1, 2, \cdots, m$, where

$$\mu_j = \frac{1}{\tau_{1j}} > 0,\ \rho_i = \frac{1}{\tau_{2i}} > 0.$$

τ_{1j} and τ_{2i} mean the integration time constants of the integrators. A new conceptual circuit implementation of the above set of differential equations is shown in Fig. 5. The circuit consists of adders (summing amplifiers) and integrators. Diodes used in the feedback from the integrators assure that the output voltages x_j are non-negative (i.e., $x_j \geqq 0$). A regularization in this circuit is performed by using local feedback with gain α around appropriate integrators. The circuit can be realized as a continuous-time or discrete-time network depending on what kind of integrators are used.

In order to realize the discrete-time network we can transform the set of differential equations into a system of difference equations:

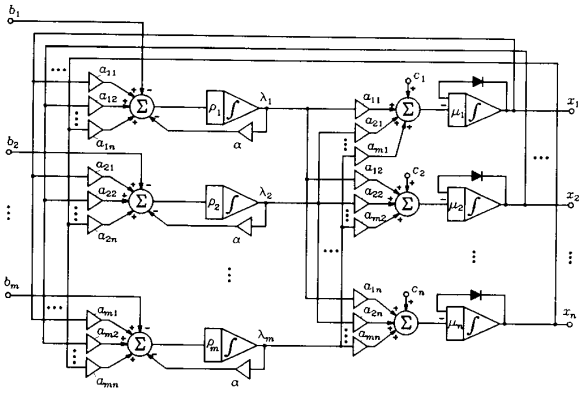


Fig. 5. A conceptual ANN implementation of linear programming.

$$\boldsymbol{x}^{(k+1)} = g_p\left[\boldsymbol{x}^{(k)} - \mu^{(k)}\left(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}^{(k)}\right)\right] \qquad (52a)$$

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho^{(k)}\left[\boldsymbol{A}\boldsymbol{x}^{(k)} - \boldsymbol{b} - \alpha\boldsymbol{\lambda}^{(k)}\right] \qquad (52b)$$

for $k = 1, 2, \cdots$; $\mu^{(k)} > 0$, $\rho^{(k)} > 0$, where

$$g_p\left(\boldsymbol{y}^{(k)}\right) := \begin{cases} \boldsymbol{y}^{(k)} & \text{if } \boldsymbol{y}^{(k)} > \boldsymbol{0}, \\ \boldsymbol{0} & \text{otherwise.} \end{cases} \qquad (52c)$$

The above system of difference equations can practically be implemented by using SC techniques [1], [5], [6]. A new SC ANN network for linear programming is shown in Fig. 6. The circuit in Fig. 6(a) consists of an SC resistance array with conductances denoted by a_{ij}, b_i, c_j and two layers of SC (discrete-time) integrators. In this circuit all the conductances are realized by using capacitors and CMOS switches as explained in Fig. 6(b). Note that this capacitor array with associated switches is shared by the first and second layer of SC integrators due to the application of the time-multiplexing technique [1]. A further simplification of the circuit shown in Fig. 6 is possible if all coefficients a_{ij} are non-negative. This condition can easily be satisfied by employing a preconditioning transformation presented in Section 3.4 (cf., (38)). Fig. 7 shows a simplified realization of the SC ANN for linear programming under the assumption that all coefficients a_{ij}, b_i, c_j are non-negative. The main advantage of the proposed circuit in comparison with other known solutions [2], [3], [6], [7] is the reduced number of connection weights (capacitors) and the improved convergence properties for ill-conditioned problems. It should be noted that the circuit performs, exactly, the matrix operation $\boldsymbol{A}^T\boldsymbol{A}$ where $\boldsymbol{A} = [a_{ij}]$. The performance of the circuits proposed in Figs. 5, 6, and 7 has successfully been computer simulated and tested for various ill-conditioned problems (cf., Section VI).

VI. Study Examples—Computer Simulation Results

To check the correctness and performance of the proposed circuit architectures we have simulated on a computer and also built all the circuit models of the ANN's considered in

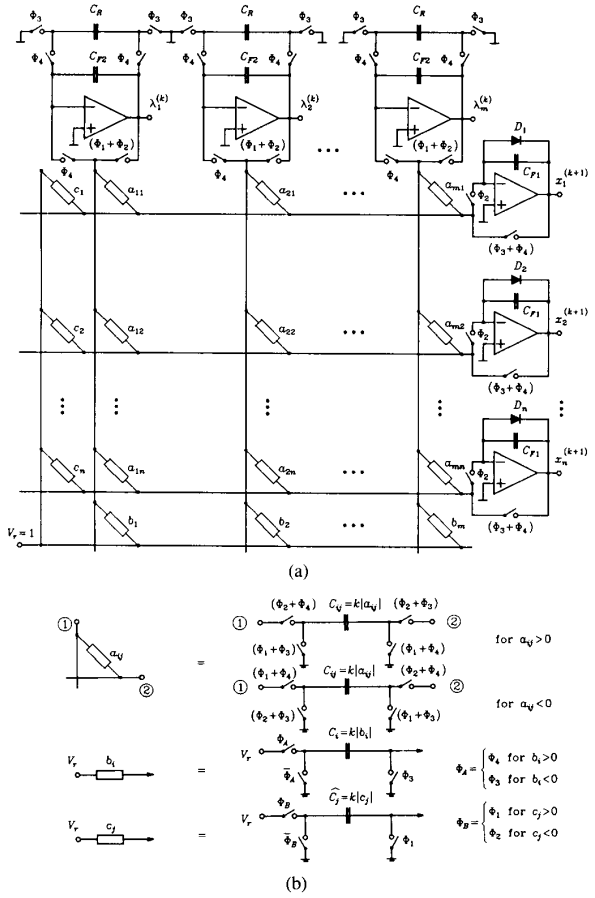


Fig. 6. (a) The linear programming SC artificial neural network. (b) Exemplary SC realizations of conductances (connection weights).

the previous sections. These models have been checked for many different numerical examples, and good agreement with the theoretical considerations have been obtained. Due to limited space we shall present in this paper only some illustrative examples.

Example 1: Let us consider the following ill-conditioned linear programming problem:

$$\min \{ \boldsymbol{c}^T \boldsymbol{x},\ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geqq \boldsymbol{0} \} \tag{53a}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 2 & -2 & 1 & 1 & 0.5 \\ 1 & 1 & 1 & 0 & 0.5 \\ 3 & -1 & 2 & 1 & 1 \end{bmatrix}$$

$$\boldsymbol{c} = [1 \quad 1 \quad 0 \quad 0 \quad 0]^T, \quad \boldsymbol{b} = [2 \quad 1 \quad 3]^T$$

$$\boldsymbol{x} = [x_1, x_2, x_3, x_4, x_5]^T.$$

Note that rank $(\boldsymbol{A}) = 2$. By use of (38) the linear constraint

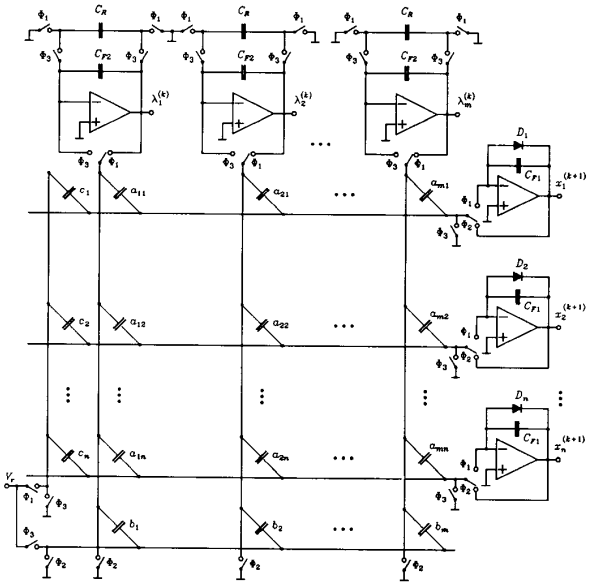


Fig. 7. The linear programming SC ANN for all connection weights non-negative.

equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is transformed to a new one with

$$\hat{\boldsymbol{A}} = \begin{bmatrix} 2 & 0 & 1 & 1 & 0.5 & 2 \\ 1 & 1 & 1 & 0 & 0.5 & 0 \\ 3 & 0 & 2 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{53b}$$

$$\hat{\boldsymbol{b}} = [2 \quad 1 \quad 3 \quad 0]^T, \quad \hat{\boldsymbol{x}} = [x_1, x_2, x_3, x_4, x_5, x_6]^T.$$

On the basis of the system of difference equations (52 a), (b) and the circuit structure of Fig. 7, an appropriate SC circuit has been designed and simulated on a computer with the following parameters:

$$\alpha = C_r / C_{F2} = 0.5, \ \mu^{(k)} = \rho^{(k)} = 100$$

and the clock frequency

$$f_c = 1\,\text{MHz}.$$

Exemplary simulation results obtained by using the TUTSIM program[7] [30] are shown in Fig. 8. The final solution (equilibrium point) was

$$\hat{\boldsymbol{x}}^* = [0 \quad 0 \quad 0.80001 \quad 0.99998 \quad 0.40002 \quad 0]^T \tag{54a}$$

which is in excellent agreement with the exact minimal norm solution

$$\boldsymbol{x}^* = [0 \quad 0 \quad 0.8 \quad 1 \quad 0.4]^T \tag{54b}$$

obtained by using a numerical program. It is worth mentioning that the circuit exhibits robustness with respect to small parameter variations, especially nonidealities (offset voltage

[7] Twente University of Technology simulation program (TUTSIM) is a fully interactive powerful program for simulation of analog continuous-time and discrete-time dynamic systems [30].

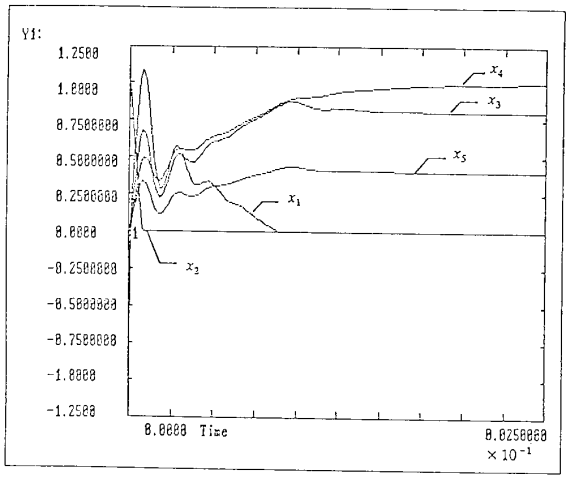


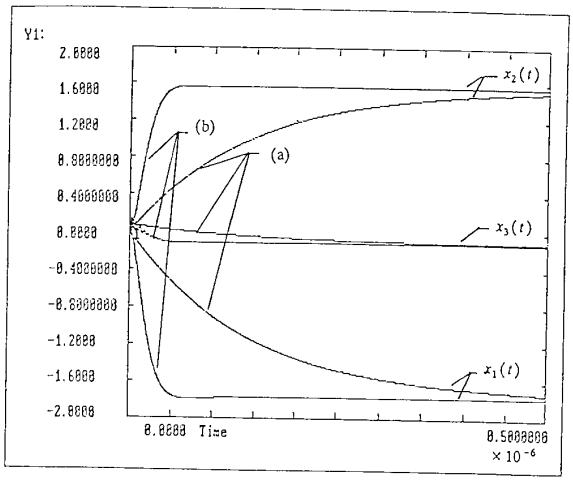
Fig. 8. Computer simulated state trajectories for the linear programming problem formulated in Example 1.

and drift) of the SC integrators. The circuit requires approximately 2000 clock cycles to solve this linear programming problem (i.e., 2 ms). In order to decrease the settling time one can increase the clock frequency and/or gains $\mu^{(k)}$ and $\rho^{(k)}$. However, if these gains are too large, the iterative algorithm (52 a), (b) may be unstable [16]. Alternatively, we can use the continuous-time algorithm (51 a), (b) (and the associated network depicted in Fig. 5) in which the gains can be arbitrarily large without causing stability problems [33].

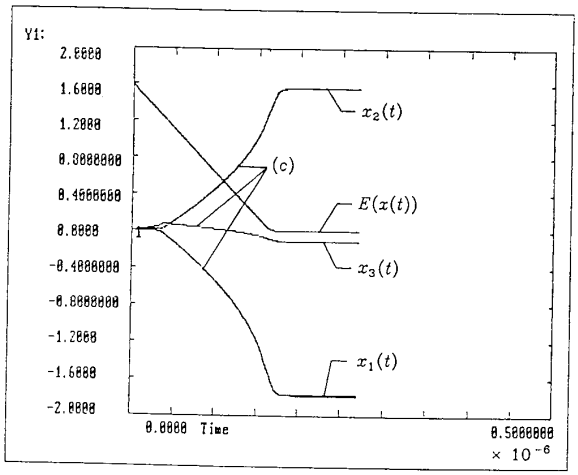
Example 2: Consider the problem of finding the solution of the system of linear equations

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{55}$$

On the basis of the general architectures shown in Figs. 1, 3, and 4 corresponding continuous-time circuits have been designed and simulated on computer. Fig. 9(a) and (b) shows computer simulated state trajectories of the variables $x_1(t)$, $x_2(t)$, and $x_3(t)$. From these results it is obvious that the structure of Fig. 3 provides superior convergence properties in comparison to the structure of Fig. 1 (cf., Fig. 9(a)). The steady-state values $\boldsymbol{x}^* = [x_1^*, x_2^*, x_3^*]^T = [-1.7777, 1.55555, -0.1111]^T$ are reached for the structure of Fig. 1 approximately in 0.75 μs while for the structure of Fig. 3 the steady state is reached in only 0.052 μs under the assumption that the time constants of the integrators equal 0.01 μs. It was found that both circuits converge to the correct solution independently of the initial starting point. Moreover, it was confirmed that the sigmoidal nonlinearities introduced into the computing units (adders) are nonessential and have no effect on the final solution since the matrix $\boldsymbol{A}$ is nonsingular. However, we can improve the convergence speed (i.e., decrease the settling time) by incorporating some nonlinearities. This fact is illustrated in the following example.



(a)



(b)

Fig. 9. Computer simulated state trajectories for the linear equation solution of Example 2. (a) The state trajectories for the circuit structure of Fig. 1. (b) The state trajectories for the circuit structure of Fig. 3. (c) The state trajectories and the energy function $E[\boldsymbol{x}(t)] = \sum_{i=1}^{3} r_i^2(\boldsymbol{x}(t))$ for the circuit structure of Fig. 4. (It has been assumed that the circuit reaches the steady-state in the time $t_r \leq 0.18\mu$s.)

Example 3: Find the inverse of the matrix

$$\boldsymbol{A} = \begin{bmatrix} 66 & 78 & 27 \\ 78 & 93 & 36 \\ 27 & 36 & 45 \end{bmatrix}. \tag{56}$$

In order to find the inverse matrix we need to make the source vector $\boldsymbol{b}$ successively $[1, 0, 0]^T$, $[0, 1, 0]^T$, $[0, 0, 1]^T$. We must repeat the computation three times to find three columns of the inverted matrix $\boldsymbol{B} = \boldsymbol{A}^{-1}$. Note that cond $(\boldsymbol{A}) = 1232.4$, and the eigenvalues of the matrix are

$$\lambda_1 = 0.1414, \lambda_2 = 29.58, \lambda_3 = 174.28.$$

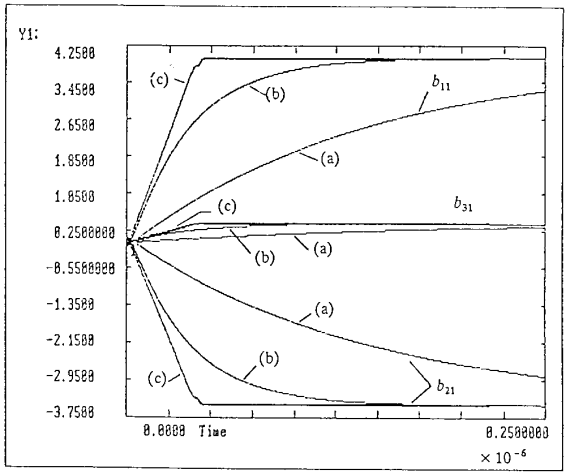


Fig. 10. Computer simulated state trajectories for the matrix inversion (Example 3). (a) The state trajectories for Model a) (cf., (57a)). (b) The state trajectories for Model b) (cf., (57b)). (c) The state trajectories for Model c) (cf., (57c)).

Since the matrix $\boldsymbol{A}$ is symmetric and positive-definite, simplified circuit models with a reduced number of elements can be used (cf., Table I). For this problem three different circuits have been designed on the basis of the following general models (cf., Table I):

$$\text{a)} \quad \frac{dx_j}{dt} = -\mu_0 r_j(\boldsymbol{x}); \ \mu_0 = \frac{1}{\tau} = 10^8 \tag{57a}$$

$$\text{b)} \quad \begin{cases} \dfrac{dx_j}{dt} = -\mu_0\left(k_j r_j(\boldsymbol{x}) + \lambda_j\right); & \mu_0 = \rho_0 = \dfrac{1}{\tau} = 10^8 \\ \dfrac{d\lambda_j}{dt} = \rho_0\left(r_j(\boldsymbol{x}) - \alpha\lambda_j\right); & \alpha = 0.5, \ k_j = 0.1 \end{cases} \tag{57b}$$

$$\text{c)} \quad \begin{cases} \dfrac{dx_j}{dt} = -\mu_0 \operatorname{sign}\left(k_j r_j(\boldsymbol{x}) + \lambda_j\right) \\ \dfrac{d\lambda_j}{dt} = \rho_0\left(r_j(\boldsymbol{x}) - \alpha\lambda_j\right) \end{cases} \tag{57c}$$

where $r_j(\boldsymbol{x}) = \sum_{i=1}^{3} a_{ji}x_i - b_j$ for $j = 1, 2, 3$.

Fig. 10 shows the computer simulated state trajectories obtained by computing the first column of the inverted matrix for the three models a), b), and c). From this figure it is evident that the circuit with "signum" nonlinearities (Model c)) provides the best convergence speed. The steady-state values are reached for this circuit in approximately 0.05 μs. In this case the total matrix inversion is obtained in less than 0.15 μs. Unfortunately, small oscillations[8] around the steady-state values were observed, which degrades the final accuracy. The best accuracy can be achieved by the circuit based on Model c). For this model the following solution was obtained after a total time approximately equal to 0.6 μs:

$$\boldsymbol{B} = \left[b_{ij}\right] = \boldsymbol{A}^{-1} = \begin{bmatrix} 3.961 & -3.480 & 0.406 \\ -3.480 & 3.073 & -0.369 \\ 0.407 & -0.368 & 0.073 \end{bmatrix}. \tag{58}$$

This is in good agreement with the exact solution.

The circuit based on Model a) is approximately 10 times slower than Model c) (i.e., the steady-state solutions are obtained in approximately 1.5 μs).

The speed of convergence may play an important role if the coefficients a_{ij} and b_j are variable in time.

Example 4: Consider the problem of linear estimation described by the overdetermined set of equations

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \\ 3 \end{bmatrix}. \tag{59}$$

It is assumed that all errors are confined to the observation vector $b = [1, 1, 2, 3, 3]^T$. On the basis of the system of differential equations (13)–(15) an appropriate circuit has been designed (cf., Fig. 1). The circuit has been simulated on the computer for various shapes of sigmoid nonlinearities (cf., (14)), i.e., different sets of parameters α and β. For example, for the parameters $\alpha = 0.2$ $\beta = 5$ the simulating circuit has found (see Fig. 11)

$$x_1^* = 0.206, \ x_2^* = 0.598 \tag{60a}$$

with the residual vector

$$r(\boldsymbol{x}^*) = \left[-0.196, 0.402, 0, -0.402, 0.196\right]^T$$

which is very close to the standard least squares (l_2-norm) estimation

$$x_1^* = 0.200, \ x_2^* = 0.600 \tag{60b}$$
$$r(\boldsymbol{x}^*) = \left[-0.2, 0.4, 0, -0.4, 0.2\right]^T.$$

It was found that the circuit is insensitive with respect to variation of the parameters α and β in the ranges $0.1 \leqq \alpha \leqq 2$, $1 \leqq \beta \leqq 10$ and also for the nonidealities of the integrators (i.e., the offset voltage and the finite dc gain of the employed op-amps). However, for the parameters $\alpha = 100$, $\beta = 1$ the circuit has found the solution (cf., Fig. 11)

$$x_1^* = 0.491, \ x_2^* = 0.501 \tag{61a}$$
$$r(\boldsymbol{x}^*) = \left[-0.008, 0.493, -0.006, -0.505, 0.004\right]^T$$

which is close to the least absolute value (l_1-norm) estimation

$$x_1^* = x_2^* = 0.5, \ r(\boldsymbol{x}^*) = \left[0, 0.5, 0, -0.5, 0\right]^T. \tag{61b}$$

It is interesting to note that the l_1 estimate is the median, while the l_2 estimate is the mean [22], [23]. Note also that

[8] The reasons for these parasitic oscillations is under current investigation.

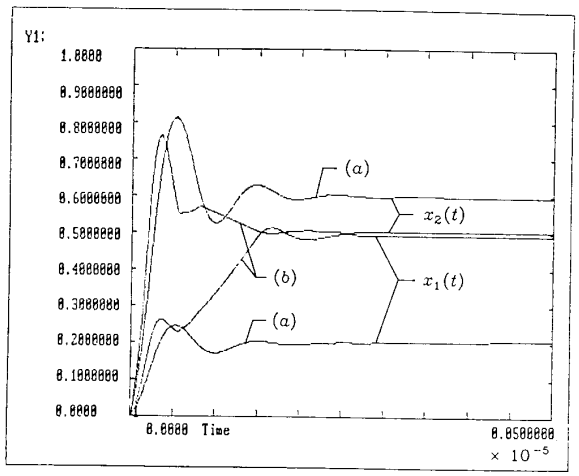


Fig. 11. Computer simulated state trajectories for Example 4 (cf., (14) and (15)). (a) The state trajectories for the model using the ordinary (linear) least-squares criterion ($\alpha = 0.2, \beta = 5$). (b) The state trajectories for the model using the iteratively reweighted least-squares criterion ($\alpha = 100$, $\beta = 1$).

the estimate given by (61a) approximates rather closely (1), (3), and (5) while almost ignoring (2) and (4), which strongly deviate from the fit. On the other hand, the residuals of the l_2 estimate (cf., (60a)) are all nonzero.

It has been found by extensive computer simulation that, in general, the l_2 estimates are strongly affected by any change in the observed vector $\boldsymbol{b}$, while the l_1 estimates are resistant to some large changes in the data [22], [33], [34]. This is a very useful property when the known data in the vector $\boldsymbol{b}$ are contaminated with occasional outliers or spiky noise.

VII. Conclusions

Potentially, the artificial neural networks represent very realistic and promising architectures for linear algebraic high-speed computing involving on-line control algorithms for large multivariable systems. In this paper, neuron-like architectures are considered for on-line high-speed computation of a system of linear equations, inversion of matrices and linear programming problems suited for VLSI fabrications. Some new structures are proposed that require a reduced number of basic computing units (cells) and/or exhibit an improved dynamic performance. New architectures of ANN's are developed on the basis of the augmented Lagrangian with regularization terms. For these architectures (cf., Fig. 3) the advantage of convergence speed has been verified in many computer simulation experiments where a consistent reduction in the settling time ranging roughly from 30% to 80% in comparison to the "standard" structure of Fig. 1 has been observed. The important advantages of such circuits is that the computation time (i.e., settling time) is principally not influenced by the size of the problem (because of the massively parallel nature of neural networks), but rather depends on the condition (measured, e.g., by the condition number or the location of the eigenvalues of the matrix $\boldsymbol{A}$) and the architecture and the parameters of the ANN. Since the currently available CMOS VLSI technology enables us to implement an ANN with a hundred or even more neurons with fully programmable synaptic weights, it is expected that the ANN discussed in this paper will be able to solve systems of linear equations with up to a few thousands of coefficients. An alternative for the VLSI CMOS implementation of the ANN, if the interconnection problem is severe, may be to use electrooptic interconnections [36], which may, in the near future, lead to the realization of practical and efficient networks for the high-speed solving of large systems of linear and nonlinear equations.

Acknowledgment

The authors would like to thank Professor G. O. Martens (University of Manitoba, Winnipeg, Canada) and the referees for many valuable comments that have greatly improved the presentation.

References

[1] R. Unbehauen and A. Cichocki, *MOS Switched-Capacitor and Continuous-Time Integrated Circuits and Systems – Analysis and Design*. New York: Springer-Verlag, 1989.

[2] D. W. Tank and J. J. Hopfield, "Simple 'neural' optimization networks: an A/D converter, signal decision circuits, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 533–541, May 1986.

[3] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 554–562, May 1988.

[4] R. P. Lippman, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, pp. 4–22, Apr. 1987.

[5] Y. P. Tsividis and D. Anastossiou, "Switched capacitor neural networks," *Electronics Lett.*, vol. 23, pp. 958–959, 1987.

[6] A. Rodrigues-Vazquez, A. Rueda, J. L. Huertas, and R. Dominiquez-Castro, "Switched-capacitor neural networks for linear programming," *Electronics Lett.*, vol. 24, pp. 496–498, 1988.

[7] P. Mars and H. R. McLean, "Implementation of linear programming with a digital stochastic computer," *Electronics Lett.*, vol. 12, pp. 516–517, 1976.

[8] S. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[9] T. Kohonen, *Self-Organization and Associative Memory*. New York: Springer-Verlag, 1988.

[10] A. D. Culhane, M. C. Peckerar, and C. R. K. Marrian, "A neural net approach to discrete Hartley and Fourier Transforms," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 695–702, May 1989.

[11] R. H. Sturges, Jr., "Analog matrix inversion," *IEEE J. Robotics Automat.*, vol. 4, pp. 157–162, Apr. 1988.

[12] D. Z. Anderson, Ed., *Neural Information Processing Systems*. New York: American Institute of Physics, 1988.

[13] G. Martinelli and R. Perfetti, "Neural network for real-time synthesis of FIR filters," *Electronics Lett.*, vol. 25, pp. 1199–1200, 1989.

[14] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. New York: Academic Press, 1981.

[15] L. F. Shampine and C. W. Gear, "A user's view of solving stiff ordinary differential equations," *SIAM Rev.*, vol. 21, pp. 1–17, 1979.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computation*. Oxford: North Oxford Academic, 1986.

[17] S. Van Huffel and J. Vandewalle, "Analysis and properties of the generalized total least squares problem $Ax \approx B$ when some or all columns in A are subject to errors," *SIAM J. Matrix Anal. Appl.*, vol. 10, pp. 294–315, 1989.

[18] P. J. Huber, *Robust Statistic*. New York: Wiley, 1981.

[19] F. R. Hampel, E. M. Ronchetti, P. Roussew, and W. A. Stahel, *Robust Statistics – The Approach Based on Influence Functions*. New York: Wiley, 1987.

[20] D. Coleman, P. Holland, N. Kaden, and V. Klema, "A system of

subroutines for iteratively reweighted least squares computations," *ACM Trans. Math. Software,* vol. 6, pp. 327-336, 1990.
[21] D. P. O'Leary, "Robust regression computation using iteratively reweighted least squares," *SIAM J. Matrix Anal. Appl.,* vol. 11, pp. 466-480, 1990.
[22] I. Barrodale, C. A. Zala, and N. R. Chapman, "Comparison of L_1 and L_2 norms applied to one-at-a-time spike extraction from seismic traces," *Geophysics,* vol. 49, pp. 2048-2052, 1984.
[23] I. Barrodale and C. A. Zala, "L_1 and L_∞ curve fitting and linear programming algorithms and applications," in *Numerical Algorithms,* J. L. Mohamed and J. E. Walsh, Eds. Oxford: Oxford University Press, 1986, ch. 11, pp. 220-238.
[24] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems.* Englewood Cliffs, NJ: Prentice-Hall, 1974.
[25] L. Elden, "Algorithms for the regularization of ill-conditioned least squares problems," *BIT,* vol. 17, pp. 134-145, 1977.
[26] T. Poggio and F. Girosi, "Network for approximation and learning," *Proc. IEEE,* vol. 78, pp. 1481-1497, 1990.
[27] J. M. Varach, "A practical examination of some numerical methods for linear discrete ill-posed problems," *SIAM Rev.,* vol. 21, pp. 100-111, 1979.
[28] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods.* New York: Academic, 1982.
[29] J. C. Platt and A. H. Barr, "Constrained differential optimization," in *Proc. Neural Information Processing Systems,* pp. 612-621, Nov. 1987.
[30] User manual of TUTSIM on IBM PC Computers Version 6.55, Meerman Automation, 7160 AC Neede, The Netherlands, 1990.
[31] A. Cichocki and R. Unbehauen, "Switched-capacitor artificial neural networks for nonlinear optimization with constraints," in *Proc. ISCAS-90 Symp.,* pp. 2809-2812, May 1990.
[32] ——, "Switched-capacitor neural networks for differential optimization," *Int. J. Circuit Theory and Applications,* to be published.
[33] ——, *Introduction to Artificial Neural Networks for Computing, Optimization, and Signal Processing.* Stuttgart: Verlag Teubner, 1991.
[34] ——, "Neural networks for solving of systems of linear equations: Part II—Minimax and least absolute value problems," submitted for publication.
[35] O. Axelsson and P. S. Vassilevski, "A survey of multilevel preconditioned iterative methods," *BIT,* vol. 29, pp. 769-793, 1989.
[36] H. J. Caulfield, J. H. Gruninger, J. E. Ludman, K. Steiglitz, H. Rabitz, J. Gelfand, and E. Tsoni, "Bimodal optical computers," *Appl. Optics,* vol. 25, pp. 3128-3131, 1986.

Andrzej Cichocki received the M. Sc. (with honors), Ph.D., and Habilitate Doctorate (Dr. Sc.) degrees all in electrical engineering from the Warsaw Technical University, in 1972, 1975, and 1982, respectively.

Since 1972, he has been with the Institute of Theory of Electrical Engineering and Electrical Measurements at the Warsaw Technical University, where he became a Full Professor in 1991. He is co-author of two books, *MOS Switched-Capacitor and Continuous-Time Integrated Circuits* and Systems (Springer Verlag) and *Neural Networks for Optimization, Computing and Signal Processing* (Teubner Verlag and Wiley). His current research interest lies in the fields of neural computing, MOS analog circuit design, and nonlinear network analysis and synthesis.

Dr. Cichocki was awarded a Research Fellowship by the Alexander von Humboldt Foundation, Germany, in 1984 and he spent this fellowship with the Lehrstuhl für Allgemeine und Theoretische Elektrotechnik at the University Erlangen-Nürnberg.

Rolf Unbehauen (M'61-SM'82-F'91) received the diploma in mathematics, the Ph.D. degree in electrical engineering, and the qualification for inauguration as academic lecture from Stuttgart University, Stuttgart, Germany, in 1954, 1957, and 1964, respectively.

From 1955 to 1966 he was a member of the Institute of Mathematics, the Computer Center, and the Institute of Electrical Engineering at Stuttgart University, where he was appointed associate professor in 1965. Since 1966 he has been Full Professor of electrical engineering at the University of Erlangen-Nürnberg, Erlangen, Germany. He has published a great number of papers on his research results. He has authored four books in German and is co-author of *MOS Switched-Capacitor and Continuous-Time Integrated Circuits and Systems* (Springer, 1989). His teaching and research interests are in the areas of network theory and its applications, system theory and electromagnetics.

Dr. Unbehauen is a Member of the Informationstechnische Gesellschaft of Germany and of URSI, Commission C: Signals and Systems.