ELSEVIER

Parallel Computing 22 (1997) 1455–1475

PARALLEL COMPUTING

Neural networks for solving linear inequality systems

A. Cichocki [1], A. Bargiela *,[2]

Department of Computing, Nottingham Trent University, Nottingham NG1 4BU, United Kingdom

Received 4 July 1995; revised 15 February 1996; accepted 25 July 1996

Abstract

In this paper a neural network approach to the on-line solution of linear inequality systems is considered. Three different techniques are discussed and for each technique a novel neural network implementation is proposed. The first technique is a standard penalty method implemented as an analog neural network. The second technique is based on the transformation of inequality constraints into equality constraints with simple bounds on the variables. The transformed problem is then solved using least squares (LS) and least absolute values (LAV) optimisation criteria. The third technique makes use of the regularised total least squares criterion (RTLS). For each technique a suitable neural network architecture and associated algorithm in the form of nonlinear differential equations has been developed. The validity and performance of the proposed algorithms has been verified by computer simulation experiments. The analog neural networks are deemed to be particularly well suited for high throughput, real time applications.

Keywords: Analog neural networks; Parallel architectures; Linear inequality systems; Stochastic gradient descent optimisation

1. Introduction

Many problems in science and technology involve solving large sets of linear inequality and or equality constraints [1,7,9,10,12,15]. Furthermore in many applications

* Corresponding author. Email: andre@doc.ntu.ac.uk.

[1] Prof. A. Cichocki is now a team leader of Artificial Brain Systems and Chemical Research (RIKEN) Hirosowa 2-1, Wako Saitama 351-01, Japan, on leave from Warsaw University of Technology, Poland, and he is a Visiting Professor in the Department of Computing at Nottingham Trent University, U.K.

[2] Prof. A. Bargiela is a Head of Real Time Telemetry Systems group in the Department of Computing at Nottingham Trent University, Burton Street, Nottingham NG1 4BU, U.K.

such as image and signal restoration, computer tomography, system identification and automatic control it is required to solve such systems in real time [1,6,14]. Unfortunately for applications requiring the solution of such systems within a fraction of a millisecond a standard digital computer cannot comply with the desired computation time or alternatively its use is too expensive. One promising approach to solving such problems is to employ artificial neural networks (ANN) [1–6,11]. The main purpose of this paper is to review known techniques involving the use of analog neural networks and to propose new algorithms which require fewer neural processing units and/or offer improved computational performance.

2. Formulation of the problem

This paper outlines a neural network approach to the solution of the following classical constraint satisfaction problem [13]

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1,2,\dots,m \tag{1}$$

and/or

$$\sum_{j=1}^{(m+q)} a_{ij} x_j = b_i, \quad i = m+1, m+2, \dots, m+q. \tag{2}$$

In the expressions (1), (2) the parameters $a_{ij} \in \mathbb{R}$ and $b_i \in \mathbb{R}$ are normally expected to define well the characteristics of the modelled problem. However, in practice, these parameters are frequently known only approximately. Consequently, the constraints satisfaction problem involving a combined set of inequalities and equations is seen as a more realistic practical problem. The solution to such a problem however may be not unique and in order to find a point-solution, an additional optimisation criterion needs to be introduced.

Another concern that needs to be addressed when developing techniques for the solution of (1) and (2) is the ill-conditioning of the mathematical expressions. Frequently the practical problems are ill-conditioned, so a small perturbation of the input data may cause a large change in the solution [13]. The research challenge is to construct a numerically stable and computationally efficient algorithm for the solution of the constraint satisfaction problem (1), (2) that will be suitable for use in real-time (on-line to physical processes).A general approach advocated in this paper is that of the reformulation of the problem (1) and (2) as an optimisation problem [1,4,6]:

$$\text{minimise} \quad f(\boldsymbol{x}) \tag{3}$$

subject to inequality constraints,

$$\boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} \leq b_i, \quad i = 1,2,\dots,m \tag{4}$$

equality constraints,

$$\boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} = b_i, \quad i = m+1, m+2, \dots, m+q$$

and simple bounds,

$$x_{j_{\min}} \le x_j \le x_{j_{\max}}, \quad j = 1,2,\dots,n, \tag{6}$$

where $x \in \mathbb{R}^n$, $a_i^{\mathrm{T}} = [a_{i1}, a_{i2}, \dots, a_{in}]$ is the ith row of a matrix $A \in \mathbb{R}^{(m+q)\times n}$ and $f(x) \in \mathbb{R}$ is the appropriately defined cost function. The cost function $f(x)$ can take various forms, e.g.:

(i) $f(x) = \text{const}$ – pure linear inequality/equality problem;
(ii) $f(x) = \frac{1}{p}\|x\|_p^p$ – p-norm problem;
(ii.a) $f(x) = \frac{1}{2}\|x\|^2$ – least squares problem;
(ii.b) $f(x) = c^{\mathrm{T}} x$ – linear programming problem;
(iii) $f(x) = \sum_{j=1}^{q} - x_j \ln x_j$, $x_j > 0$ – maximum entropy problem.

In this paper, without loss of generality, in order to simplify notations, the optimisation problem (3) with inequality constraints (4) and simple bounds (6) will be considered.

3. Analog neural network models using penalty function approach

The mapping of a constrained optimisation problem into an appropriate loss (cost) function is a standard approach in the design of artificial neural networks (ANN) [2–6]. Consequently the construction of an appropriate energy function $E(x)$ for which the lowest energy state will correspond to the optimal solution x^* is a pre-requisite for the formulation of the optimisation problem in terms of ANN.

For the minimisation problem (3), (4), (6), a general energy function, based on the penalty method [6,14] can be constructed:

$$E(x) = f(x) + k\sum_{i=1}^{m} v(r_i(x)) \tag{7a}$$

or equivalently

$$E(x) = \nu f(x) + \sum_{i=1}^{m} v(r_i(x)), \tag{7b}$$

where $k > 0$ denotes the penalty parameter, $\nu > 0$ is the reciprocal penalty parameter, $r_i(x) = a_i^{\mathrm{T}} x - b_i$ are the residuals, and $v(r_i(x)) = \max\{0, \mathrm{sgn}(r_i(x)) P(r_i(x)\}$ are the penalty functions with $P(r_i(x)) \ge 0$.

For example, the penalty functions may take one of the following forms:

$$\text{(a)} \quad P(r) = \tfrac{1}{2} r^2 \quad \text{(quadratic)}, \tag{8a}$$

$$\text{(b)} \quad P(r)|r| \quad (l_1 - \text{norm}), \tag{8b}$$

$$\text{(c)} \quad P(r) = \begin{cases} r^2/2 & \text{for } |r| \le \beta \\ \beta|r| - \beta^2/2 & \text{for } |r| > \beta \end{cases} \quad \text{(Hubers)}, \tag{8c}$$

$$\text{(d)} \quad P(r) = \beta^2 \ln\cosh(r/\beta), \ \beta > 0 \quad \text{(logistic)}. \tag{8d}$$

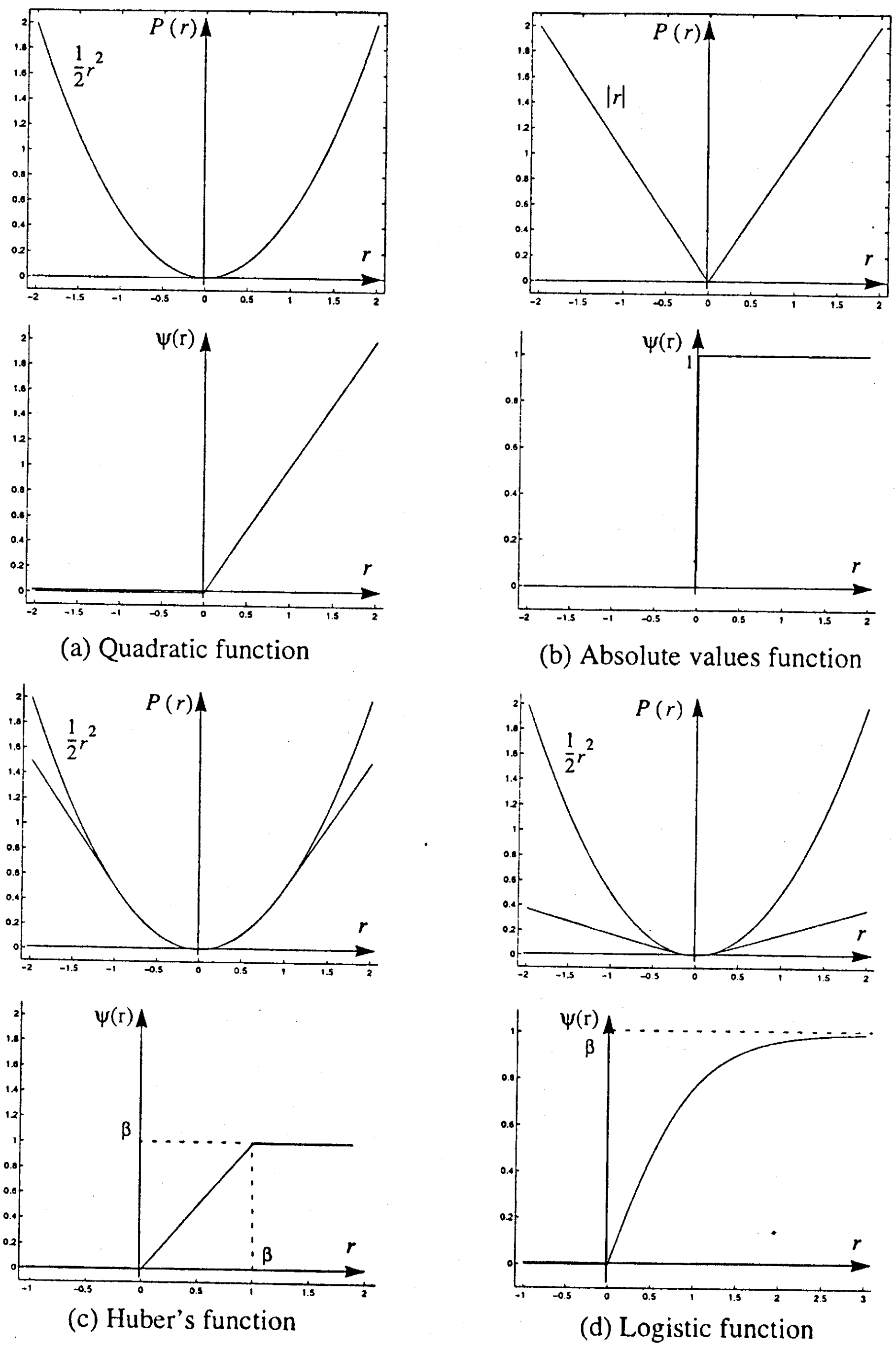


Fig. 1. Penalty functions. (a) Quadratic function. (b) Absolute values function. (c) Huber's function. (d) Logistic function.

The plots of these functions and the associated first derivatives,

$$\Psi(r)=\begin{cases}P'(r) & \text{for } r\geq 0\\ 0 & \text{for } r<0\end{cases} \tag{9}$$

are shown in Fig. 1(a)–(d).

It is known from the optimisation theory [14] that, except for trivial cases, only non-differentiable penalty functions $P(r_i)$ enable an exact solution to the original constrained optimisation problem in a single unconstrained minimisation while assuming a finite value of the penalty parameter k. Usually, in order to ensure a feasible solution satisfying all of the inequality constraints, the penalty parameter k in (7a) must tend to infinity. This is rather inconvenient from the implementation point of view. Therefore, it is proposed to use equation (7b) in which the parameter $\nu>0$ decreases to zero as time increases to infinity. Often a compromise is accepted by setting the parameter $\nu=$ const to a sufficiently small value, so that the resulting solution can be very close to the exact unconstrained optimisation problem [6].

Using the standard gradient descent method for the minimisation of the energy function $E(x)$ the optimisation problem (7b) can be mapped onto a system of nonlinear ordinary differential equations

$$\frac{dx_j}{dt}=-\mu_j\frac{\partial}{\partial x_j}E(x)=-\mu_j\left[\nu\varphi_j(x_j)+\sum_{i=1}^{m}a_{ij}\Psi(r_i)\right],\quad j=1,2,\ldots,n, \tag{10}$$

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

$$\varphi_j(x_j)=\frac{\partial}{\partial x_j}f(x) \tag{11}$$

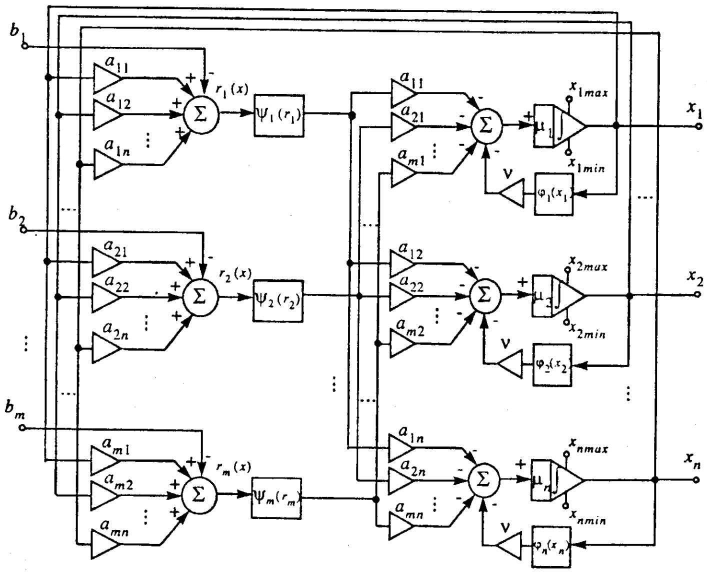
activation function of the output neurons,

$$\Psi_i(r_i)=\begin{cases}P'(r_i) & \text{if } r_i\geq 0,\\ 0 & \text{otherwise,}\end{cases} \tag{12}$$

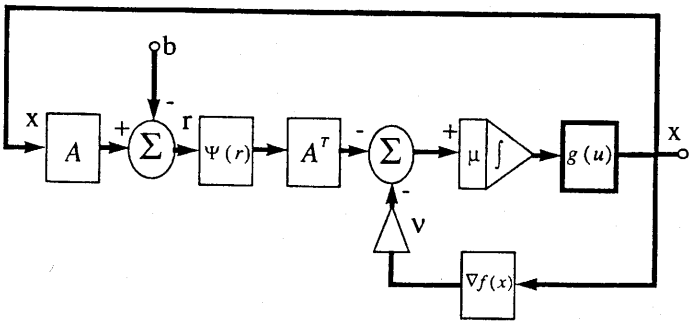
activation function of input neurons.

On the basis of the set of differential equations (10)–(12) the ANN with suitable connection weights, the activation functions $\varphi_j(x_j)$ and $\Psi_i(r_i)$ and input excitations has been designed. The detailed functional block diagram of the ANN implementing these equations is given in Fig. 2(a). The network consists of limiting integrators, adders (summing amplifiers) with associated connection weights a_{ij} and nonlinear building blocks implementing the activation functions (see Fig. 1 and Fig. 3). The network of Fig. 2(a) consists of two layers of processing units. The first layer calculates the residuals $r_i(x)$ and the errors $\Psi(r_i(x))$, while the variables of interest x_j are calculated in the second layer which combines and integrates in time the errors $\Psi(r_i)$.

It should be noted that simple bounds constraints on state variables: $x_{j_{\min}}\leq x_j\leq x_{j_{\max}}$ (cf. (6)) are conveniently implemented using integrators with signal limiters at their output. The input signals are integrated but they cannot drive the output beyond the specified limits $[x_{j_{\min}},x_{j_{\max}}]$. In such an approach, all simple bounds constraints are "hard", i.e., the constraints must not be violated either at the final solution or during the



(a) Detailed neural network



(b) Aggregated neural network

Fig. 2. Neural network using penalty function approach. (a) Detailed neural network. (b) Aggregated neural network.

optimisation process. Equivalently one can employ a nonlinear transformation which maps an unlimited output signal u_j into an output-limited signal x_j of the jth integrator,

$$x_j = g_j(u_j), \tag{13}$$

e.g.

$$x_j = x_{j_{\min}} + \frac{x_{j_{\max}} - x_{j_{\min}}}{1 + e^{-\gamma u_j}}, \tag{14}$$

$\gamma > 0.$

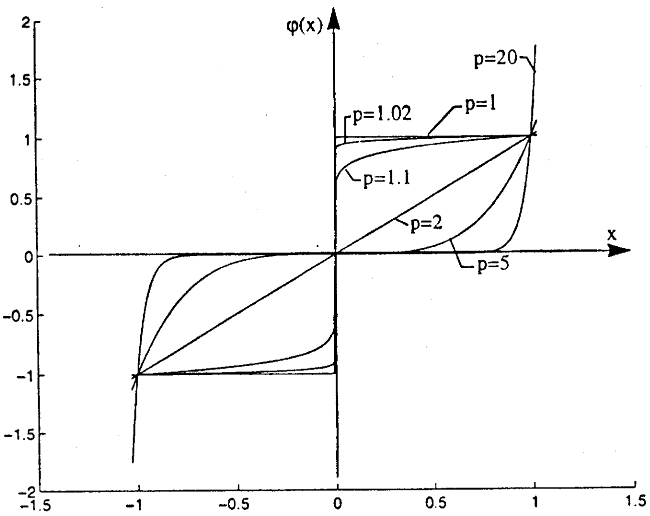


Fig. 3. Exemplary plot of the activation function $\varphi(x_j) = (\partial/\partial x_j)f(x)$ with $f(x) = (1/p)\|x\|_p^p$.

The artificial neural network making use of the above transformations is presented in Fig. 2(b) and it is described by a system of differential equations as follows:

$$\frac{d}{dt}u = -\hat{\mu}[\nabla f(x) + A^{\mathrm{T}}\Psi(r(x))]), \tag{15}$$

$$x = g(u), \tag{16}$$

where

$$\hat{\mu} = \operatorname{diag}\{\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_n\}, \quad \mu_i > 0,$$

$$g(u) = [g_1(u_1), g_2(u_2), \ldots, g_n(u_n)]^{\mathrm{T}},$$

$$r(x) = Ax - b,$$

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^{\mathrm{T}}. \tag{17}$$

It should be noted that due to employing appropriate activation functions in the output layer the satisfaction of the bound constraints is ensured.

4. Neural network models using transformation of inequality constraints into equality constraints

The neural network models can be considerably simplified if all inequality constraints of the form

$$a_i^{\mathrm{T}} x \leq b_i \tag{18}$$

are converted into equality constraints.

It is easy to note that the above set of inequality constraints can be re-written as follows to satisfy the requirement that $b_i \geq 0$ for all i:

$$a_i^{\mathrm{T}} x \leq b_i, \quad i = 1,2,\ldots,k, \tag{19}$$

$$a_i^{\mathrm{T}} x \geq b_i, \quad i = k+1,\ldots,m. \tag{20}$$

Such inequality constraints can be transformed into corresponding linear equations by means of an auxiliary variable $y_i \geq 0$, [3],

$$a_i^{\mathrm{T}} x - b_i y_i = 0, \tag{21}$$

with $0 \leq y_i \leq 1$ for $i = 1,2,\ldots,k$ and $y_i \geq 1$ for $i = k+1,\ldots,m$.

The minimisation problem (3), (4) can now be re-formulated as follows:

$$\text{minimise} \quad f(x) \tag{22}$$

subject to equality constraints

$$Ax - b_{\mathrm{D}} y = 0 \tag{23}$$

and simple bounds

$$x_{j_{\min}} \leq x_j \leq x_{j_{\max}}, \tag{24}$$

$$0 \leq y_i \leq 1 \quad \text{for } i = 1,2,\ldots,k, \tag{25}$$

$$y_i \geq 1 \quad \text{for } i = k+1,\ldots,m, \tag{26}$$

where

$$A = [a_1, a_2, \ldots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n},$$

$$b_{\mathrm{D}} = \operatorname{diag}\{b_1, b_2, \ldots, b_m\} \in \mathbb{R}^m,$$

$$y \in \mathbb{R}^m, \quad x \in \mathbb{R}^n.$$

For this problem the following energy function can be formulated,

$$E(x,y) = \nu f(x) + \sum_{i=1}^{m} P[r_i(x,y)], \tag{27}$$

where $\nu > 0$, $r_i(x,y) = a_i^{\mathrm{T}} x - y_i b_i$ and $P(r_i)$ are penalty function terms as specified in (8a)–(8d).

Minimisation of the energy function according to the gradient descent method leads to a system of differential equations

$$\frac{d}{dt} x = -\mu \left[\nabla f(x) + A^{\mathrm{T}} \Psi(Ax - b_{\mathrm{D}} y) \right], \tag{28}$$

$$\frac{d}{dt} y = \mu b_{\mathrm{D}} \overline{\Psi}(Ax - b_{\mathrm{D}} y), \tag{29}$$

with simple bounds (24)–(26), where

$$\mu = \operatorname{diag}\{\mu_1, \mu_2, \ldots, \mu_n\}, \quad \mu_j > 0 \text{ for all } j,$$

$$\overline{\Psi}(Ax - b_{\mathrm{D}} y) = \overline{\Psi}(r(x,y)) = \left[\frac{\partial P}{\partial r_1}, \frac{\partial P}{\partial r_2}, \ldots, \frac{\partial P}{\partial r_m} \right]^{\mathrm{T}} \tag{30}$$

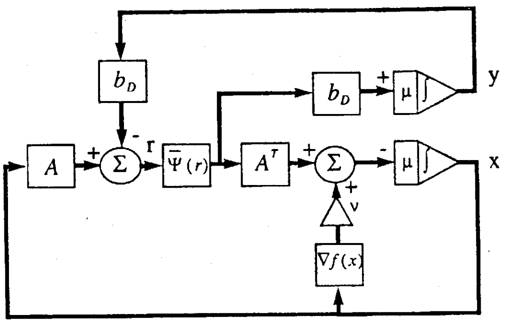


Fig. 4. Neural network implementing transformation of inequality constraints into equality constraints.

A functional block diagram illustrating the implementation of the algorithm described by (28)–(29) is shown in Fig. 4.

It is interesting to note that in a special case of $P(r_i) = (1/2)r_i^2$, for all i, the above algorithm simplifies as follows

$$\frac{d}{dt}x = -\mu\left[-\nabla f(x) + (A^{\mathrm{T}}A)x - \left(A^{\mathrm{T}}b_{\mathrm{D}}\right)y\right], \tag{31}$$

$$\frac{d}{dt}y = \mu\left[b_{\mathrm{D}}Ax - b_{\mathrm{D}}^2 y\right], \tag{32}$$

with simple bounds (24)–(26)

An alternative conversion of inequalities (19), (20) into equations is as follows. The "k" inequality constraints (19) can be replaced by

$$a_i^{\mathrm{T}}x + x_{n+i} = b_i, \tag{33}$$

where $x_{n+i} \geq 0$ is a new variable called a slack variable. Analogously, the remaining "$m-k$" inequality constraints (20) can be replaced by

$$a_i^{\mathrm{T}}x - x_{n+i} = b_i, \tag{34}$$

where $x_{n+i} \geq 0$ is a new variable called a surplus variable.

Using (33) and (34) the optimisation problem (3), (4) is expressed as follows

$$\text{minimise} \quad f(x) \tag{35}$$

subject to equality constraints

$$\vec{A}\hat{x} = b \tag{36}$$

and simple bounds

$$x_{j_{\min}} \leq x_j \leq x_{j_{\max}} \quad \text{for } j = 1,2,\ldots,n, \tag{37}$$

$$x_j \geq 0 \quad \text{for } j = n+1,\ldots,n+m, \tag{38}$$

where $\hat{x} = [x, x_{n+1},\ldots,x_{n+m}]^{\mathrm{T}}$, $\vec{A} = \left[A, \vec{I}_m\right] \in \mathbb{R}^{m\times(n+m)}$ and $\vec{I}_m$ is a diagonal matrix with the first k elements equal 1 and the rest of diagonal elements equal -1.

For the above problem the following energy function can be formulated

$$E(\hat{x}) = \nu\vec{f}(\hat{x}) + \sum_{i=1}^{m} P(r_i(\hat{x})), \tag{39}$$

where $\vec{f}(\hat{x}) = [f(\hat{x}), O]^{\mathrm{T}}$ and O is a vector of m coordinates equal 0.

Minimisation of the energy function (39) according to gradient descent method leads to a system of differential equations:

$$\frac{d\hat{x}}{dt} = -\mu\left[\nu\nabla\vec{f}(\hat{x}) + \vec{A}^{\mathrm{T}}\Psi\left(\vec{A}\hat{x} - b\right)\right], \tag{40}$$

where

$$\Psi\left(\vec{A}\hat{x} - b\right) = \Psi(\hat{r}(\hat{x})) = \left[\frac{\partial}{\partial\hat{r}_1}P(\hat{r}), \ldots, \frac{\partial}{\partial\hat{r}_m}P(\hat{r})\right]^{\mathrm{T}}$$

and $\vec{\nabla}(\hat{x}) = [\nabla f(\hat{x}), O]^{\mathrm{T}}$ and O is a vector of m coordinates equal 0.

A special but very important case is when $P(\hat{r}_i(\hat{x})) = |\hat{r}_i(\hat{x})|$, for all i, where $\hat{r}(\hat{x}) = \sum_{j=1}^{n+m}\hat{a}_{ij}x_j - b_j$. In this case the energy function contains the sum of the absolute values of the residuals so the criterion is called the least absolute values criterion (LAV). Minimising such an energy function leads to the set of the following differential equations:

$$\frac{d\hat{x}}{dt} = \mu\left\{\nu\nabla\vec{f}(\hat{x}) + \vec{A}^{\mathrm{T}}\mathbf{sgn}[\hat{r}(\hat{x})]\right\}, \tag{41}$$

where $\hat{r}(\hat{x}) = \vec{A}\hat{x}$, $\mathbf{sgn}(\hat{r}) = [\mathrm{sgn}(r_i), \ldots, \mathrm{sgn}(r_m)]$and

$$\mathrm{sgn}(\hat{r}_i) = \begin{cases} 1, & \hat{r}_i > 0, \\ -1, & \hat{r}_i < 0. \end{cases}$$

The algorithm based on this equation is called the LAV algorithm. A characteristic feature of this algorithm is its ability to reject erroneous data so the solutions are spanned by the most correct data while the outliers are ignored. However, the penalty function terms $P(\hat{r}_i) = |\hat{r}_i|$ are not differentiable at $r_i = 0$ so a modified criterion approximating the LAV criterion and differentiable at $r_i = 0$ can be introduced as follows:

$$P(\hat{r}_i) = \beta \ln\cosh(\hat{r}_i/\beta) \quad \beta > 0 \tag{42}$$

In this case the set of differential equations takes the form

$$\frac{d\hat{x}}{dt} = -\mu\left\{\nu\nabla\vec{f}(\hat{x}) + \vec{A}^{\mathrm{T}}\mathbf{tanh}[(\hat{r}(\hat{x}))/\beta]\right\}, \tag{43}$$

where$\hat{r}(\hat{x}) = \vec{A}\hat{x} - b$, $\mathbf{tanh}(\hat{r}/\beta) = [\tanh(\hat{r}_1/\beta), \ldots, \tanh(\hat{r}_m/\beta)]^{\mathrm{T}}$.

Note that for a very small β the function $\tanh(r)$ closely approximates the $\mathrm{sgn}(r)$ function so the solution to (43) is the LAV solution. However, for large β, $\beta \gg 1$, the

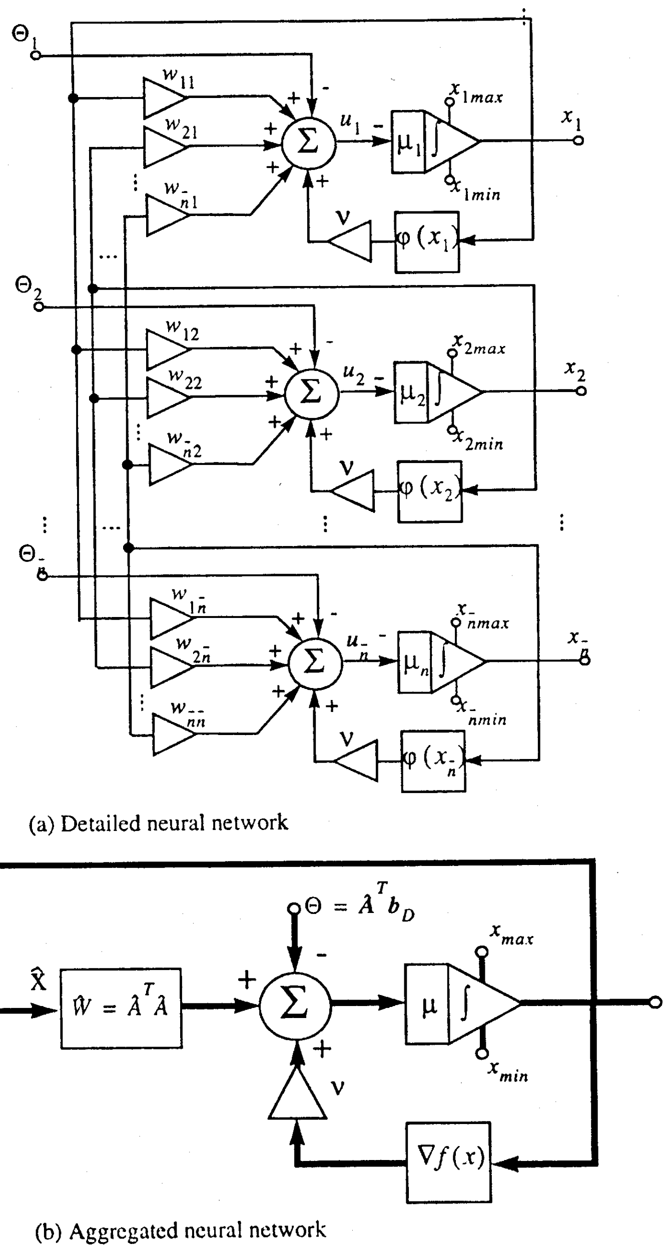


Fig. 5. Simplified neural network implementing transformation of inequality constraints into equality constraints. (a) Detailed neural network. (b) Aggregated neural network.

function tanh(r) approximates a linear function in an arbitrarily large range, so the solution to (43) approximates a regularised least squares solution:

$$\frac{d\hat{x}}{dt} = -\mu\left[\nu\vec{\nabla f}(\hat{x}) + \vec{A}^{\mathrm{T}}\left(\vec{A}\hat{x} - b\right)\right]. \tag{44}$$

The same equation can be obtained by directly applying the criterion $P(\hat{r}_i) = \frac{1}{2}\hat{r}^2$. Equation (44) can be simplified as follows:

$$\frac{d\hat{x}}{dt} = -\mu\left[\nu\vec{\nabla f}(\hat{x}) + W\hat{x} - \Theta\right], \tag{45}$$

where $W = \vec{A}^{\mathrm{T}}\vec{A} \in \mathbb{R}^{\bar{n}\times\bar{n}}$, $\Theta = \vec{A}^{\mathrm{T}}b \in R^{\bar{n}}$ and $\bar{n} = n + m$.

The neural network implementation of the simplified differential equation (45) is shown in Fig. 5.

It should be pointed out that the number of processing units (neurons) can be reduced by combining the two processing layers. However, this necessitates some pre-processing of data and it may be inconvenient for large matrices, particularly, when the entries a_{ij} and/or b_i are varying with time.

The network architectures discussed so far require a rather large number of processing units (analog multipliers and adders). What follows is a discussion of another approach which makes it possible to solve the regularised LS problem using only a single neuron with adaptive synaptic weights.

As a preliminary to further discussion the following instantaneous residuals function is considered:

$$\tilde{r}[\hat{x}(t)] = s^{\mathrm{T}}\hat{r} = \sum_{i=1}^{m} s_i(t)\hat{r}_i(\hat{x}(t)), \tag{46}$$

where $s^{\mathrm{T}} = [s_1(t), s_2(t), \ldots, s_m(t)]$ is a vector of zero-mean, uncorrelated, identically distributed external excitation signals [1] and

$$\hat{r} = \vec{A}\hat{x} - b.$$

It is worth noting that with the above definition of s^{T} the residuals function is equal to zero if and only if the constraint $\vec{A}\hat{x} = b$ is exactly satisfied. Elaborating the equation (46) one can write

$$\tilde{r}[\hat{x}(t)] = \sum_{j=1}^{n} \tilde{a}_j(t)\hat{x}_j(t) - \tilde{b}(t) + \sum_{i=1}^{k} s_i(t)x_{i+n}(t) - \sum_{i=k+1}^{m} s_i(t)x_{i+n}(t), \tag{47}$$

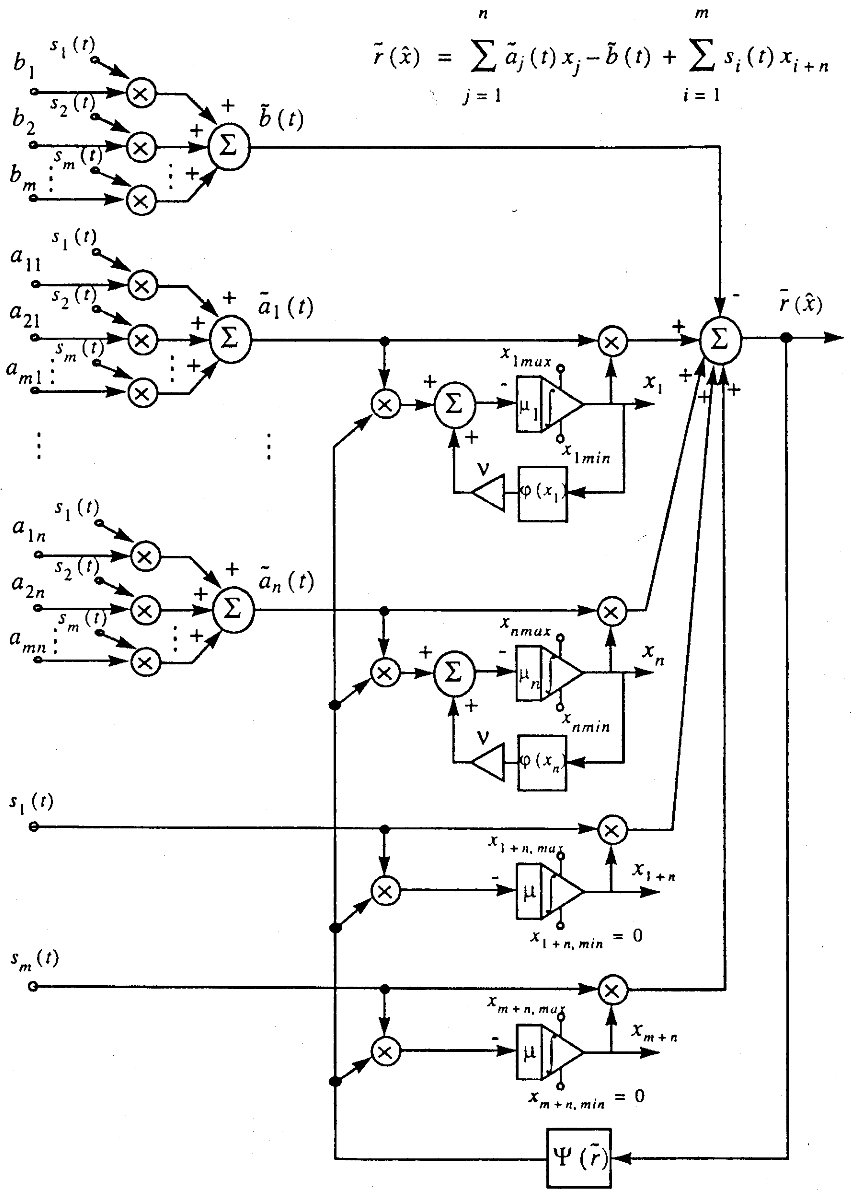
where $\tilde{a}_j(t) = \sum_{i=1}^{m} a_{ij}s_i(t)$ and $\tilde{b}(t) = \sum_{i=1}^{m} b_i s_i(t)$.

Using the above instantaneous residuals function the instantaneous estimate of the energy (cost) function at time t can be defined as follows

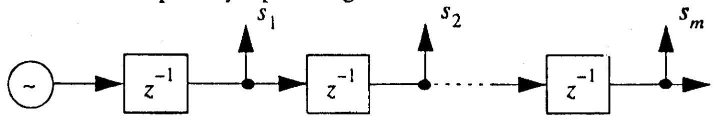
$$E(\hat{x}(t)) = \nu f(x) + P[\tilde{r}(\hat{x}(t))], \tag{48}$$

where $P(\tilde{r})$ is the penalty function term, e.g. one given by equations (8a)–(8d).

Fig. 6. Neural network using random perturbation signal. (a) Neuron with adaptive synaptic weights. (b) Unit delays.



(a) Neuron with adaptive synaptic weights



(b) Unit delays

Minimisation of the energy leads to a system of differential equations

$$\frac{d}{dt}x_j(t) = -\mu_j\left[\nu\varphi\left(x_j(t)\right) + \tilde{a}_j(t)\Psi\left[\tilde{r}(\hat{x}(t))\right]\right], \tag{49}$$

$$\frac{d}{dt}x_{i+n}(t) = -\mu s_i(t)\Psi\left[\tilde{r}(\hat{x}(t))\right], \tag{50}$$

for $j = 1,2,\ldots,n$ and $i = 1,2,\ldots,m$ and where

$$\mu > 0, \quad \nu > 0, \quad \varphi\left(x_j(t)\right) = \frac{\partial}{\partial x_j}f(\hat{x}(t)), \quad \Psi\left[\tilde{r}(\hat{x}(t))\right] = \frac{\partial}{\partial \tilde{r}}P(\tilde{r}).$$

In a special case of $P(\tilde{r}) = \frac{1}{2}\tilde{r}^2$ the system of differential equations (49)–(50) simplifies to

$$\frac{d}{dt}x_j(t) = -\mu_j\left[\nu\varphi\left(x_j(t)\right) + \tilde{a}_j(t)\tilde{r}(\hat{x}(t))\right], \tag{51}$$

$$\frac{d}{dt}x_{i+n}(t) = -\mu s_i(t)\tilde{r}[\hat{x}(t)], \tag{52}$$

for $j = 1,2,\ldots,n$ and $i = 1,2,\ldots,m$.

The systems of differential equations (49)–(50) and (51)–(52) represent basic adaptive learning algorithms for a single artificial neuron as shown in Fig. 6. The network is driven by the products of the incoming data a_{ij} and b_i and the zero-mean uncorrelated pseudo-random signals $s_i(t)$. It is self-evident that the artificial neuron shown in Fig. 6 allows concurrent processing of information. If only one pseudo-random signal generator is available, the m excitation signals $s_i(t)$ can be generated using a chain of unit delays as shown in Fig. 6(b).

It should be noted that the neural network can be further simplified by replacing the analog multipliers, which can be quite expensive, with simple switches or sign reversers controlled by a pseudo-random multiphase binary generator [6].

5. Neural network model using regularised total least squares criterion

The criteria considered so far assumed that all errors are confined to the observation vector $\boldsymbol{b}$ and that the data matrix A is free from errors. However, such an assumption is often unrealistic since the sampling errors, modelling errors and instrument errors imply possible inaccuracies of the entries in matrix A. The total least squares criterion (TLS) has been introduced to reflect this more realistic approach to the estimation problem [6,16].

We consider now the optimisation problem (3)–(6) with $q > n$. This problem can be transformed to the following:

$$\text{minimise} f(\boldsymbol{x})\, \boldsymbol{x} \in \mathbb{R}^n \tag{53}$$

subject to equality constraints

$$\vec{A}\hat{\boldsymbol{x}} = \boldsymbol{b} \tag{54}$$

and simple bounds

$$x_{j_{\min}} \le x_j \le x_{j_{\max}}, \quad j = 1,2,\ldots,n, \tag{55}$$

$$x_{i+n} \ge 0, \quad i = 1,2,\ldots,m, \tag{56}$$

where $b \in \mathbb{R}^{m+q}$, $\hat{x} = [x_1, x_2, \ldots, x_{n+m}] \in \mathbb{R}^{n+m}$, $\vec{A} \in \mathbb{R}^{(m+q)\times(n+m)}$.

Note that the system of linear equations $\vec{A}\hat{x} = b$ is overdetermined (since $(m+q) > (n+m)$) and a cost function $f(x)$ can play a role of a regularisation function.

Using the regularised total least squares criterion the following energy function can be constructed

$$E_{\mathrm{RTLS}}(\hat{x}(t)) = \nu f(x) + \sum_{i=1}^{\overline{m}} \left(\frac{\hat{r}_i(\hat{x}(t))}{\sqrt{1 + \hat{x}^{\mathrm{T}}\hat{x}}} \right)^2, \tag{57}$$

where $\overline{m} = m + q$ and $\hat{r}_i(\hat{x}(t)) = \sum_{i=1}^{n} \hat{a}_{ij} x_j(t) - b_i$.

The first term $\nu f(x)$ is the regularisation term whose purpose is to force a smoothness constraint on the estimated optimal solution x° for ill-conditioned problems. The regularisation parameter $\nu \ge 0$ determines the relative importance of this term. The second term is the standard TLS term [6] which forces the weighted sum of residuals

$$\sum_{i=1}^{\overline{m}} \left(\frac{\hat{r}_i(\hat{x}(t))}{\sqrt{1 + \hat{x}^{\mathrm{T}}\hat{x}}} \right)^2$$

to be minimal.

Comparing to the standard least squares, the solution of the total least squares problem is computationally quite burdensome. This is probably why the TLS method has not been used in optimisation as widely as it might be expected.

We propose now a new simple adaptive learning algorithm that overcomes the problem of computational complexity of the TLS.

For this purpose we formulate the following instantaneous energy function:

$$E_{\mathrm{RTLS}}(\hat{x}(t)) = \nu f(x) + \frac{1}{2} \frac{\tilde{r}^2[\hat{x}(t)]}{\hat{x}^{\mathrm{T}}(t)\hat{x}(t) + 1} \tag{58}$$

where

$$\tilde{r}_i(\hat{x}(t)) = \sum_{i=1}^{n} \tilde{a}_j(t) x_j(t) - \tilde{b}(t) + \sum_{i=1}^{k} s_i(t) x_{i+n}(t) - \sum_{i=k+1}^{m} s_i(t) x_{i+n}(t)$$

,$\hat{r}(t) = \vec{A}\hat{x}(t) - b$, $\tilde{a}_j(t) = \sum_{i=1}^{\overline{m}} a_{ij} s_i(t)$, and $\tilde{b}(t) = \sum_{i=1}^{\overline{m}} b_i s_i(t)$.

Applying now a standard gradient descent algorithm we obtain a set of differential equations

$$\frac{d}{dt} x_j(t) = -\mu(t) \left\{ \nu \frac{\partial}{\partial x_j} f(x) + \tilde{r}(t) \frac{\tilde{a}_j(t)(\hat{x}^{\mathrm{T}}\hat{x} + 1) - \tilde{r}(t) x_j(t)}{(\hat{x}^{\mathrm{T}}\hat{x} + 1)^2} \right\} \tag{59}$$

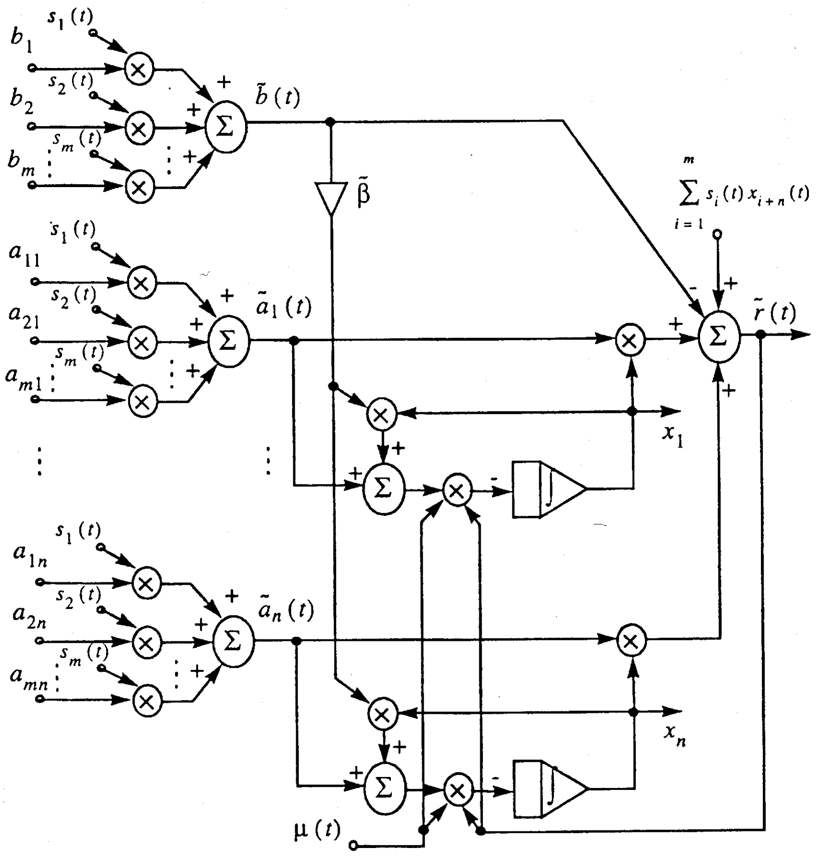


Fig. 7. Generalised network implementing the regularised total least squares (RTLS).

and

$$\frac{d}{dt}x_{i+n}(t) = -\mu(t)s_i(t)\tilde{r}(t), \tag{60}$$

where $\mu(t) > 0$ and is typically decreasing to zero.

The above system of differential equations simplifies, after linearisation, to the following

$$\frac{d}{dt}x_j(t) = -\mu(t)\left\{\nu\varphi(x_j) + \tilde{r}(t)\left[\tilde{a}_j(t) + \tilde{b}(t)x_j(t)\right]\right\} \tag{61}$$

for $j = 1,2,\ldots,n$ and

$$\frac{d}{dt}x_{i+n}(t) = -\mu(t)s_i(t)\tilde{r}(t) \tag{62}$$

for $i = 1,2,\ldots,m$, which in effect is an adaptive parallel learning algorithm for solving the regularised total least squares (RTLS) problem.

A diagram illustrating the implementation of the algorithm is shown in Fig. 7. By comparing this diagram with that of Fig. 6, it can be noted that the algorithm (61)–(62)

represents a generalisation of the "standard" LS algorithm described by equations (49)–(50).

By introducing a parameter $\tilde{\beta}$ in the equation (61) a general form of the algorithm is obtained which caters for both LS and TLS criteria

$$\frac{d}{dt}x_j(t) = -\mu(t)\left\{\nu\varphi(x_j) + \tilde{r}(t)\left[\tilde{a}_j(t) + \tilde{\beta}\tilde{b}(t)x_j(t)\right]\right\}, \tag{63}$$

where $\tilde{\beta} = 0$ for LS criterion and $\tilde{\beta} = 1$ for TLS criterion. Increasing the value of the parameter $\tilde{\beta}$ even further has an effect of giving greater emphasis on errors in matrix $\vec{A}$ compared to errors associated with vector $\boldsymbol{b}$. In an extreme case of $\tilde{\beta} \gg 1$ it can be assumed that the vector $\boldsymbol{b}$ is almost free from errors and all errors are associated with entries of data matrix $\vec{A}$. Such a case is referred to in the literature, as the data least squares (DLS) problem [8].

6. Computational examples

In order to verify the correctness and performance of the proposed neural networks in solving the systems of inequalities, extensive simulation studies of the systems presented in Figs. 2 and 4–7 have been undertaken. The empirical study included a number of inequality systems as well as various penalty functions. The following is a representative sample of this study.

A system of inequalities has been considered

$$\begin{aligned} &x_1 + x_2 \leq 1.1, \\ &-x_1 - x_2 \leq -0.9, \\ &x_1 + 5x_2 \leq 3.1, \\ &-x_1 - 5x_2 \leq -2.9, \\ &-x_1 - 2x_2 \leq -2.0. \end{aligned}$$

Using the penalty function approach the above inequalities have been mapped onto a set of nonlinear ordinary differential equations, (10), which are then implemented as a neural network illustrated in Fig. 2. The penalty function adopted in this example is a sigmoidal function $P(r) = \beta^2 \ln\cosh(r/\beta)$.

The least absolute values (LAV) and the least squares (LS) solutions to the problem have been obtained by adjusting the value of the parameter β in the penalty function. The least absolute values solution found by the neural network for $\beta = 0.01$ was

$$\boldsymbol{x} = [0.5992, 0.4982]^{\mathrm{T}}$$

with the residual vector

$$\boldsymbol{r} = [-0.0026, -0.1974, -0.0097, -0.1903, 0.4043]^{\mathrm{T}}.$$

This solution compares quite well with the standard LAV solution

$$\boldsymbol{x} = [0.6000, 0.5000]^{\mathrm{T}}, \qquad \boldsymbol{r} = [0.0, -0.2, 0.0, -0.2, 0.4]^{\mathrm{T}},$$

particularly if it is noted that the neural network delivers this result extremely rapidly.

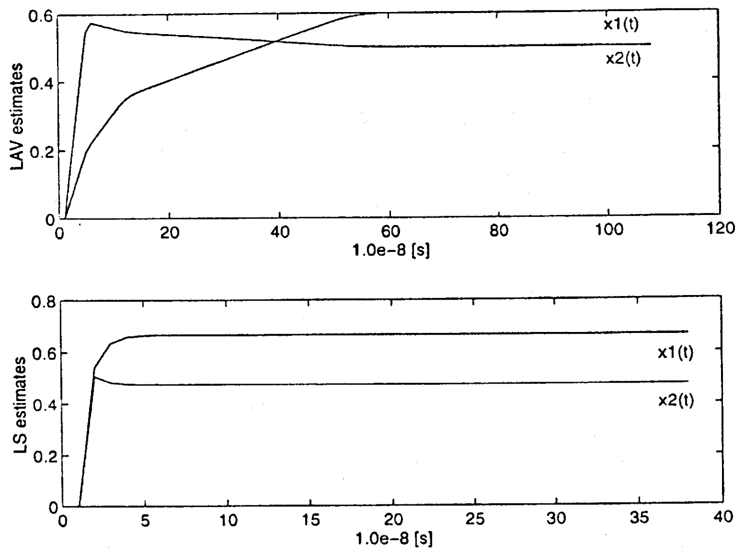


Fig. 8. MV and LS estimates obtained using the neural network of Fig. 2.

Assuming time constants of integrators of the order of 0.01 μs the steady state is reliably reached within 1 μs of the simulated time as illustrated in Fig. 8.

The least squares solution to the problem found for the parameter $\beta = 100$ was

$$x = [0.6666, 0.4762]^T$$

with the residual vector

$$r = [0.0428, -0.2428, -0.0524, -0.1476, 0.3810]^T$$

which is almost identical to the analytical LS solution

$$x = [0.6667, 0.4762]^T, \qquad r = [0.0429, -0.2429, -0.0524, -0.1476, 0.3810]^T.$$

Again the neural network converged to the solution very rapidly, as illustrated in Fig. 8, delivering the result in less than 0.1 μs of the simulated time.

The robustness of the LAV and LS solutions was investigated by varying the value of the parameter β over a range of 0.001 to 1000. It has been found, through the simulations, that the neural network converged to an accurate LAV solution for $\beta \leq 0.02$ and to a LS solution for $\beta \geq 2$. Fig. 9 illustrates the relationship between the final solution and the value of the parameter β. It is interesting to note that for the penalty function $P(r) = \beta^2 \ln\cosh(r/\beta)$,the convergence becomes significantly slower as the parameter β decreases to 0.001 (line (a) on the third diagram in Fig. 9).

To investigate the effect of the penalty function on the neural network solution, a similar test was carried out for a modified penalty function $P(r) = \beta \ln\cosh(r/\beta)$. It is quite remarkable to see that the solutions obtained for this penalty function are virtually identical to the ones obtained before; with a small variance occurring only in the transitory stage between LAV and LS solutions (lines (a) and (b) on the top two

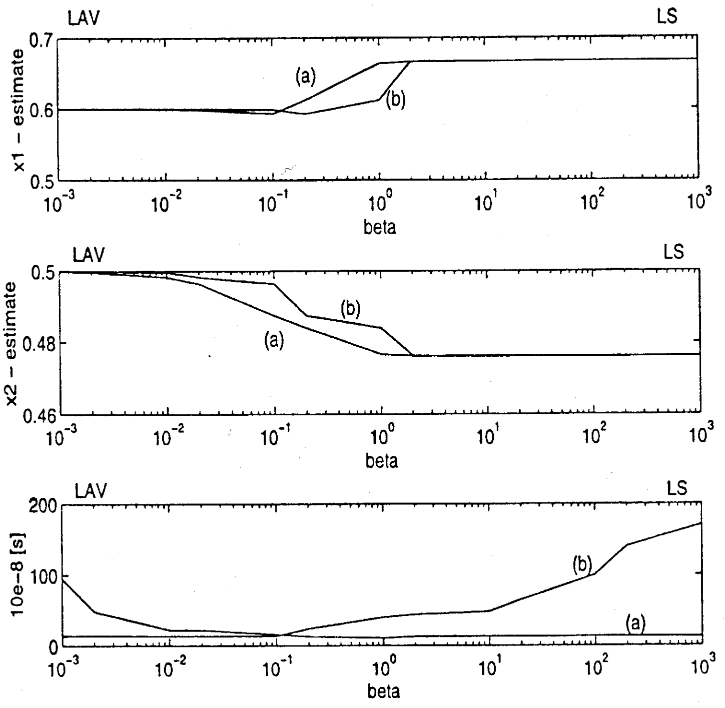


Fig. 9. Dependence of the neural network solution on the value of the parameter β in the penalty function and the effect of using different penalty functions: (a) Eq. (8d) and (b) Eq. (48).

diagrams in Fig. 9). It is interesting to note, however, that for this penalty function the convergence to the LAV solution has been improved (line (b) on the third diagram in Fig. 9). This is attributed to the fact that the corresponding activation function $\Psi(r) = \tanh(r/\beta)$ maintains a maximum and minimum value of 1 and -1 throughout the range of the parameter β.

While the simulation studies of the neural network of Fig. 2 have confirmed its excellent performance characteristics, hardware implementation of such a network is likely to be costly due to a large number of analog multipliers and adders that are needed to build it. In order to put this investigation in a correct perspective, a simplified network consisting of only one neuron with adaptive synaptic weights (Fig. 6) has been simulated and evaluated against the previous results. The penalty function used was $P(r) = \beta 2 \ln \cosh(r/\beta)$ giving an activation function $\Psi(r) = \beta \tanh(r/\beta)$ The least absolute values solution to the system of inequalities, found for the parameter $\beta = 0.01$, was

$$x = [0.5957, 0.4897]^{T}$$

and the least squares solution found for $\beta = 10$ was

$$x = [0.6679, 0.4731]^{T}.$$

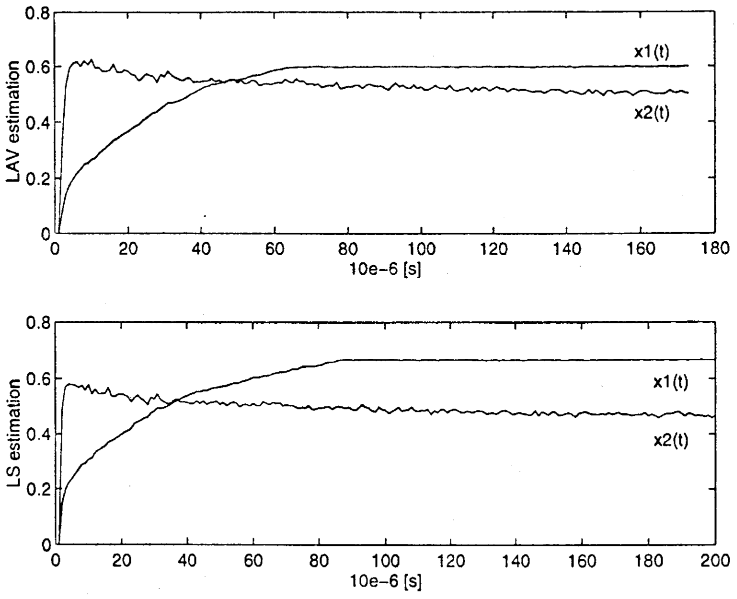


Fig. 10. LAV and LS estimates obtained using the neural network of Fig. 6.

The time trajectories of the estimates are presented in Fig. 10. It can be noticed that both LAV and LS solutions converge into a 2% envelope in approximately 0.15 ms of simulated time.

While the estimates calculated using this neural network clearly provide coarser approximation of the analytical solutions, it is evident that the network of Fig. 6 can be realised at a fraction of the cost of the network presented in Fig. 2.

It must be noted however that the above simulation experiments are intended to quantify the performance of actual analog neural networks and DO NOT suggest that large sets of inequalities should be solved by means of simulating such networks on serial computers. Indeed, the CPU times on SUN Sparc 10 workstation running MATLAB simulation software ranged from 0.1 s for the network of Fig. 2 to 800 s for the network of Fig. 6. Moreover, while the time needed by the analog neural networks is independent of the size of the problem (due to the parallel nature of processing), the CPU times required for the simulation increase with the increase of the number of inequalities that are represented by such networks.

7. Conclusions

This paper discussed the analog neural networks methodology for solving linear inequality and/or equality systems in real-time. The paper reviewed known methods and presented original techniques for the solution of this problem based on the regularised least squares (LS), least absolute values (LAV) and total least squares (TLS) approaches. Furthermore, the paper presented a new, simple and efficient adaptive learning algorithm in the form of nonlinear differential equations. It has been demonstrated that these equations can be implemented using only a single neuron with adaptive weights. The proposed networks are constructed from simple analog elements such as adders, multipliers, switches, dampers, function generators and limiting integrators.

The proposed algorithms can also be implemented on digital computers converting the systems of differential equations into appropriate sets of difference equations.

The algorithms are deemed to be particularly well suited for real-time and/or high throughput rate applications. It has been demonstrated that the analog neural networks offer a several orders of magnitude improvements in efficiency over the more conventional digital neural nets.

References

[1] S. Abe, Theories on the Hopfield neural networks with uncertainty constraints, in: *Proc. IJCNN-90*, Washington, DC, Vol. 1 (1990) 349–352.

[2] S. Amari, Mathematical theory of neural learning, *Proc. IEEE* 78 (9) (1990) 1443–1463.

[3] S. Amari, Mathematical theory of neural learning, *New Generation Comput.* 8 (1991) 281–294.

[4] S. Amari, Backpropagation and stochastic gradient descent method, *Neurocomputing* 5 (1993) 185–196.

[5] A. Bargiela, High performance neural optimisation for real time pressure control, in: *Proc. HPC-Asia Conf.*, 1994.

[6] A. Cichocki and R. Unbehauen, *Neural Networks for Optimisation and Signal Processing* (Wiley-Teubner, Chichester, UK, 1994).

[7] A. Dax, The l_1 solution of linear equations subject to linear constraints, *SIAM J. Science Stat. Comput.* 10 (2) (1989) 328–340.

[8] R.D. DeGroat and E.M. Dowling, The data least squares problem and channel equalisation, *IEEE Trans. on Signal Processing* 41 (1993) 407–411.

[9] B. Gabrys and A. Bargiela, Neural simulation of water systems for efficient state estimation, in: *Proc. ESM'95*, Society for Computer Simulation, Prague (1995) 775–779.

[10] C. Holzbaur, A specialised, incremental solved form algorithm for systems of linear inequalities, Tech. Rept. TR-94-07, Austrian Research Institute for Artificial Intelligence, University of Vienna.

[11] J.J. Hopfield and D.W. Tank, Neural computation of decision in optimising problems, *Biological Cybernetics* 52 (1985) 141–152.

[12] J.-L. Imbert, J. Cohen, M.-D. Weeger, An algorithm for linear constraint solving, *J. Logic Programming* 16 (3/4) (1993) 235–253.

[13] M. Kovacs, Stable embedding of ill-posed linear equality and inequality systems into fuzzified systems, *Fuzzy Sets and Systems* 45 (1992) 305–312.

[14] W.E. Lillo, S. Hui and S.H. Zak, Neural networks for constrained optimisation problems, *Internat. J. Circuit Theory and Appl.* 21 (4) (1993) 385–399.

[15] A. Orden, On the solution of linear equation/inequality systems, *Mathematical Programming* 1 (1971) 137–152.

[16] S. Van Huffel and J. Vandervalle, *The Total Least Squares Problem: Computational Aspects and Analysis*, Frontiers in Applied Mathematics, Vol. 9 (SIAM, Philadelphia, 1991).