



An efficient simplified neural network for solving linear and quadratic programming problems

Hasan Ghasabi-Oskoei^a, Nezam Mahdavi-Amiri^{b,*}

^a *Mathematics and Informatics Research Group, Academic Center for Education Culture
and Research, Tarbiat Modarres University,*

P.O. Box 14115-343, Tehran, Iran

^b *Department of Mathematical Sciences, Sharif University of Technology,*

P.O. Box 11365-9415, Tehran, Iran

Abstract

We present a high-performance and efficiently simplified new neural network which improves the existing neural networks for solving general linear and quadratic programming problems. The network, having no need for parameter setting, results in a simple hardware requiring no analog multipliers, is shown to be stable and converges globally to the exact solution. Moreover, using this network we can solve both linear and quadratic programming problems and their duals simultaneously. High accuracy of the obtained solutions and low cost of implementation are among the features of this network. We prove the global convergence of the network analytically and verify the results numerically.

© 2005 Elsevier Inc. All rights reserved.

Keywords: Neural network; Quadratic programming; Linear programming; Global convergence

* Corresponding author.

E-mail addresses: hgoskoei@modares.ac.ir (H. Ghasabi-Oskoei), nezamm@sina.sharif.edu (N. Mahdavi-Amiri).

1. Introduction

Solving linear and quadratic programming problems of large size is considered to be one of the basic problems encountered in operations research. In many applications a real time solution of linear or quadratic programming problem is desired. Also, most optimization problems with nonlinear objective functions are usually approximated by second order models and solved numerically by a quadratic programming technique [4,5]. Traditional algorithms such as simplex algorithm or Karmarkar's method for solving linear programming problems are computationally too expensive. One possible alternative approach is to employ neural networks on the basis of analog circuits [1–3]. The most important advantages of the neural networks are their massively parallel processing capacity and fast convergence properties.

In 1986, Tank and Hopfield [3] proposed a neural network for solving linear programming problems which was mapped onto a closed-loop circuit. Although the equilibrium point of Tank and Hopfield network may not be a solution of the original problem, this seminal work has inspired many researchers to investigate other neural networks for solving linear and nonlinear programming problems (see [6–17]). Kennedy and Chua [11] extended the Tank and Hopfield network by developing a neural network for solving nonlinear programming problems, by satisfaction of the Karush–Kuhn–Tucker optimality conditions [12]. The network proposed by Kennedy and Chua contains a penalty parameter. Thus, it generates approximate solutions only and implementation problems arise when the penalty parameter is large. To avoid the use of penalty parameters, significant work has been carried out in recent years [7,9,10]. For example, Rodriguez-Vazquez et al. [13] proposed a switched-capacitor neural network for solving a class of nonlinear convex programming problems. This network is suitable only for cases in which the optimal solutions lie within the feasible region. Otherwise, the network may have no equilibrium point [14]. Although the model proposed in [15] overcomes the aforementioned drawbacks and is robust for both continuous and discrete-time implementations, but still, the main disadvantage of the network is the requirement to use plenty of rather expensive analog multipliers for variables. Thus, not only the cost of the hardware implementation is very expensive, but also accuracy of solutions is greatly affected. The network of Xia [16,17] is an improvement over the proposal in [15] in terms of accuracy and implementation cost. The network we discuss here will be both more efficient and less costly than Xia et al.'s [15–17].

The paper is organized as follows. In Section 2, we introduce the basic problem and the model for the new neural network. Section 3, discusses some theoretical aspects of the model and analyzes its global convergence. The circuit implementation of the new model and a comparative analysis are given in

Section 4. Simulation results are shown in Section 5. Conclusions are given in Section 6.

2. Basic problems and neural network models

Consider the QP problem of the form:

$$\begin{aligned} \text{Minimize} \quad & Q(x) = \frac{1}{2}x^T Ax + c^T x, \\ \text{Subject to} \quad & Dx = b, \\ & x \geq 0 \end{aligned} \tag{1}$$

and its dual:

$$\begin{aligned} \text{Maximize} \quad & \hat{Q}(x, y) = b^T y - \frac{1}{2}x^T Ax, \\ \text{Subject to} \quad & D^T y \leq \nabla Q(x), \end{aligned} \tag{2}$$

where $\nabla Q(x) = Ax + c$ and A is an $m \times m$ real symmetric positive semidefinite matrix, D is an $n \times m$ real matrix, $y, b \in \mathfrak{R}^n$, and $x, c \in \mathfrak{R}^m$. Clearly the LP problem in standard form and its dual are special cases of the QP problem and its dual for which $A = 0_{m \times m}$.

In [15], the following neural network model was proposed for solving problems (1) and (2):

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = - \left\{ \begin{array}{l} \beta(-D^T y + Ax + c) + \beta A[x - (x + D^T y - Ax - c)^+] + D^T(Dx - b) \\ \beta\{Dx - b + D[(x + D^T y - Ax - c)^+ - x]\} \end{array} \right\}, \tag{3}$$

where $(x, y) \in \Omega$, $\Omega = \{(x, y) | y \in \mathfrak{R}^n, x \in \mathfrak{R}^m, x \geq 0\}$, $(x)^+ = [(x_1)^+, \dots, (x_m)^+]^T$, $\beta = \|x - (x + D^T y - Ax - c)^+\|_2^2$, and $(x_i)^+ = \max\{0, x_i\}$, for $i = 1, \dots, m$.

The authors [15] show that the system trajectories starting from any given initial point in Ω will converge to the solutions of the corresponding problems (1) and (2) [15].

Here, we present a new neural network model for solving problems (1) and (2) as follows:

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = - \left\{ \begin{array}{l} (I + A)[x - (x + D^T y - Ax - c)^+] \\ D[(x + D^T y - Ax - c)^+] - b \end{array} \right\}, \tag{4}$$

where $x \in \mathfrak{R}^m$, $y \in \mathfrak{R}^n$ and I is a unit matrix.

In the following section, we prove the global convergence of the new system trajectories.

3. Global convergence

In this section, we show that the new neural network described by (4) is globally convergent. We first discuss some needed results.

Theorem 1. For any $x(0) \in \mathfrak{R}^m$, $y(0) \in \mathfrak{R}^n$ there is a unique solution $z(t) = (x(t), y(t))$ with $z(0) = (x(0), y(0))$ for (4).

Proof. Let

$$F(z) = \left\{ \begin{array}{l} (I + A)[x - (x + D^T y - Ax - c)^+] \\ D[(x + D^T y - Ax - c)^+] - b \end{array} \right\}. \tag{5}$$

Note that $(x)^+$ is Lipschitz continuous. Then it is easy to see that $F(z)$ is also Lipschitz continuous. From the existence result of ordinary differential equations [18], there exists a unique solution $z(t)$ with $z(0) = (x(0), y(0))$ for (4) on some interval $[0, T]$. \square

Theorem 2. Let $\Omega^* = \{z = (x, y) \mid x \text{ solves problem (1) and } y \text{ solves problem (2)}\}$. Then $(x, y) \in \Omega^*$ if and only if (x, y) satisfies

$$\left\{ \begin{array}{l} Dx = b, \\ x = (x + D^T y - Ax - c)^+. \end{array} \right. \tag{6}$$

Proof. By the Karush–Kuhn–Tucker theorem for convex programming problem [4] we know that $(x, y) \in \Omega^*$ if and only if (x, y) satisfies

$$\left\{ \begin{array}{ll} Dx = b, \quad x \geq 0 & \text{(Primal Feasibility Condition),} \\ x^T(D^T y - Ax - c) = 0 & \text{(Complementary Slackness Condition),} \\ D^T y - Ax - c \leq 0 & \text{(Dual Feasibility Condition).} \end{array} \right. \tag{7}$$

It is easy to see that (7) is equivalent to (6). \square

Lemma 1. Let $\mathfrak{R}_+^m = \{x \mid x \geq 0\}$ and $\hat{x} \in \mathfrak{R}_+^m$. Then for any $x \in \mathfrak{R}^m, y \in \mathfrak{R}^n$ we have:

$$[(x - Ax + D^T y - c) - (x - Ax + D^T y - c)^+]^T [(x - Ax + D^T y - c)^+ - \hat{x}] \geq 0. \tag{8}$$

Proof. Since \mathfrak{R}_+^m is a closed convex set and $\hat{x} \in \mathfrak{R}_+^m$, we know by the property of a projection on a closed convex set [4] that for any $v \in \mathfrak{R}^m$, $[v - (v)^+]^T [(v)^+ - \hat{x}] \geq 0$. By setting $v = (x - Ax + D^T y - c)$ we can obtain (8). \square

We will now state and prove a lemma which serves as the basis for proving the global convergence of our model as proposed by (4).

Lemma 2. *Let $F(z)$ be as defined in (5) and let $z^* \in \Omega^*$ be fixed. Then for any $y \in \mathfrak{R}^n$, $x \in \mathfrak{R}^m$, and $z = (x, y)$ we have:*

$$(z - z^*)^T F(z) \geq \|x - (x - Ax + D^T y - c)\|_2^2. \tag{9}$$

Proof. For any $x \in R^m$, $y \in R^n$ we have:

$$\begin{aligned} (z - z^*)^T F(z) &= \left[\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x^* \\ y^* \end{pmatrix} \right]^T F(z) \\ &= \begin{pmatrix} x - x^* \\ y - y^* \end{pmatrix}^T \begin{pmatrix} (I + A)(x - (x + D^T y - Ax - c)^+) \\ D[(x + D^T y - Ax - c)^+] - b \end{pmatrix} \\ &= (x - x^*)^T (I + A)(x - (x + D^T y - Ax - c)^+) \\ &\quad + (y - y^*)^T D[(x + D^T y - Ax - c)^+] - (y - y^*)^T b \\ &= (x - x^*)^T (x - (x + D^T y - Ax - c)^+) \\ &\quad + (x - x^*)^T A(x - (x + D^T y - Ax - c)^+) \\ &\quad + [(x + D^T y - Ax - c)^+]^T (D^T y - D^T y^*) - y^T b + y^{*T} b. \end{aligned}$$

Note that

$$\begin{aligned} (x - x^*)^T [x - (x + D^T y - Ax - c)^+] &= \|x - (x + D^T y - Ax - c)^+\|_2^2 \\ &\quad + [(x + D^T y - Ax - c)^+ - x^*]^T \\ &\quad \times [x - (x + D^T y - Ax - c)^+] \end{aligned}$$

and

$$\begin{aligned} &[(x + D^T y - Ax - c)^+ - x^*]^T [x - (x + D^T y - Ax - c)^+] \\ &= [(x + D^T y - Ax - c)^+ - x^*]^T [(x + D^T y - Ax - c)^+ \\ &\quad - (x + D^T y - Ax - c)^+] + [(x + D^T y - Ax - c)^+ - x^*]^T \\ &\quad \times (Ax - D^T y + c). \end{aligned}$$

Thus by Lemma 1, we have:

$$\begin{aligned} &[(x + D^T y - Ax - c)^+ - x^*]^T [(x + D^T y - Ax - c)^+ \\ &\quad - (x + D^T y - Ax - c)^+] \geq 0. \end{aligned}$$

So,

$$\begin{aligned} & (x - x^*)^T [x - (x + D^T y - Ax - c)^+] \\ & \geq \|x - (x + D^T y - Ax - c)^+\|_2^2 + [(x + D^T y - Ax - c)^+ - x^*]^T \\ & \quad \times [Ax - D^T y + c] \end{aligned}$$

and

$$\begin{aligned} (z - z^*)^T F(z) & \geq \|x - (x + D^T y - Ax - c)^+\|_2^2 + [(x + D^T y - Ax - c)^+ - x^*]^T \\ & \quad \times [Ax - D^T y + c] + [(x + D^T y - Ax - c)^+ - x]^T A(x^* - x) \\ & \quad + [(x + D^T y - Ax - c)^+]^T (D^T y - D^T y^*) - y^T b + y^{*T} b \\ & = \|x - (x + D^T y - Ax - c)^+\|_2^2 + [(x + D^T y - Ax - c)^+]^T \\ & \quad \times [Ax - D^T y + c + Ax^* - Ax + D^T y - D^T y^*] - (x^*)^T \\ & \quad \times [Ax - D^T y + c] - x^T A(x^* - x) - b^T y + b^T y^* \\ & = \|x - (x + D^T y - Ax - c)^+\|_2^2 + [(x + D^T y - Ax - c)^+]^T \\ & \quad \times [Ax^* - D^T y^* + c] - (x^*)^T Ax + (x^*)^T D^T y - (x^*)^T c \\ & \quad - x^T Ax^* + x^T Ax - b^T y + b^T y^*. \end{aligned}$$

Knowing that $b^T y^* - c^T x^* = (x^*)^T Ax^*$ and $Dx^* = b$, we will have:

$$\begin{aligned} (z - z^*)^T F(z) & \geq \|x - (x + D^T y - Ax - c)^+\|_2^2 + [(x + D^T y - Ax - c)^+]^T \\ & \quad \times [Ax^* - D^T y^* + c] - (x^*)^T Ax + (x^*)^T Ax^* - x^T Ax^* + x^T Ax \\ & = \|x - (x + D^T y - Ax - c)^+\|_2^2 + [(x + D^T y - Ax - c)^+]^T \\ & \quad \times [Ax^* - D^T y^* + c] + (x - x^*)^T A(x - x^*). \end{aligned}$$

Since $(u(x, y))^+ \geq 0$ and x^* , (x^*, y^*) are optimal solutions of primal and dual problems, respectively, and A is a positive semidefinite matrix, we then conclude:

$$(z - z^*)^T F(z) \geq \|x - (x + D^T y - Ax - c)^+\|_2^2. \quad \square$$

Theorem 3. Let $\Omega^0 = \{z = (x, y) | F(z) = 0\}$. Then $\Omega^0 = \Omega^*$.

Proof. Let $z \in \Omega^0$. Then $F(z) = 0$. By Lemma 2 and model (4) we know that $Dx = b$ and $x = (x + D^T y - Ax - c)^+$, and thus $z \in \Omega^*$ by Theorem 2. So $\Omega^0 \subset \Omega^*$. Conversely, let $z \in \Omega^*$. Then by Theorem 2 we know from (6) that $F(z) = 0$. Thus $z \in \Omega^0$ and $\Omega^* \subset \Omega^0$. Therefore $\Omega^0 = \Omega^*$. \square

We now prove the following important result.

Theorem 4. Assume that the set Ω^* is nonempty. Then the neural network described by (4) is globally convergent to the exact solutions of the corresponding problems (1) and (2).

Proof. Let $z(0) = (x(0), y(0))$ be an initial point taken in \mathfrak{R}^{n+m} and let $z(t)$ be the solution of the initial value problem associated with (4). Then by Lemma 2 we have:

$$\frac{d}{dt} \|z(t) - z^*\|_2^2 = (z(t) - z^*)^T \frac{dz(t)}{dt} = -(z(t) - z^*)^T F(z(t)) \leq 0 \quad \forall t \geq 0,$$

where $x^* \in \Omega^*$ is fixed. Thus $\|z(t) - z^*\|_2^2 \leq \|z(0) - z^*\|_2^2, \forall t \geq 0$, and hence the solution $z(t)$ for (4) is bounded. So $z(t)$ can be uniquely extended to the infinite time interval $[0, +\infty)$. The rest of the proof can now be completed straightforwardly. \square

Remark 1. From Theorem 4 we see that the convergence region of the new model (4) is \mathfrak{R}^{n+m} , in contrast to Ω , that of the existing model (3), proposed in [15].

4. Circuit implementation of the new model and a comparison

For convenience, let $r = (x + D^T y - Ax - c)^+$. Then our proposed model (4) and the model (3) proposed in [15] are respectively represented as:

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} (I + A)(r - x) \\ -Dr + b \end{bmatrix}, \tag{10}$$

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} -\beta(D^T y - Ax - c) + \beta A(r - x) - \lambda \\ -\beta(Dr - b) \end{bmatrix}. \tag{11}$$

A block diagram of model (10) is shown in Fig. 1, where the vectors c, b are the external inputs, and the vector x, y are the network outputs. A conceptual artificial neural network (ANN) implementation of the vector r is shown in Fig. 2, where $A = (g_{ij})$ and $D = (d_{ij})$.

Remark 2. We note that the neural network corresponding to (10) is much simpler than the one corresponding to (11), and has no need for any analog multiplier. Our network is also preferential to the simplified neural network proposed in [17], since it is less complicated in terms of the required hardware, has no need for λ processing of [17], and, as we will see in the next section, produces more accurate solutions.

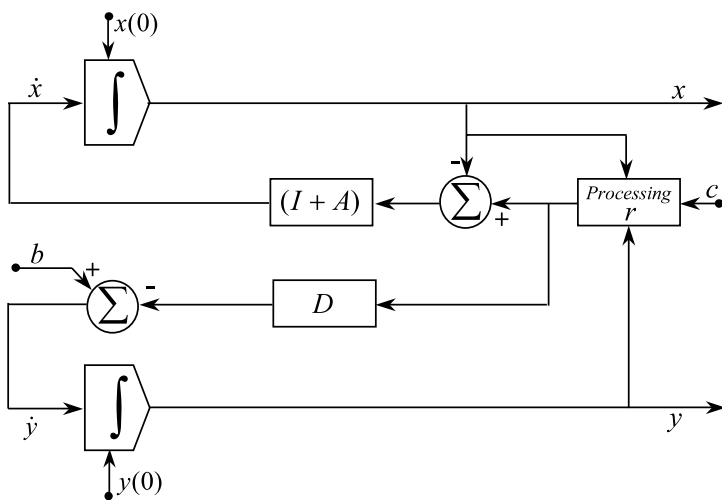


Fig. 1. A simplified block diagram of model (10).

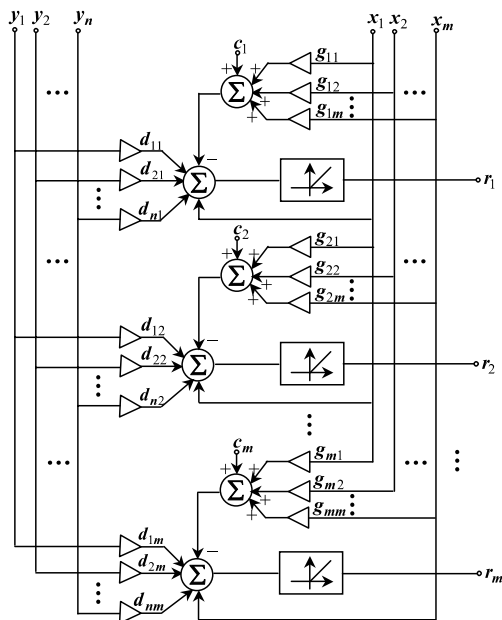


Fig. 2. Block diagram of processing r .

Remark 3. We can consider linear programming problems by setting $A = 0_{m \times m}$ in the quadratic model. Hence, by incorporating this observation, our new

neural network model can be easily used for solving linear programming problems.

5. Simulation examples

We discuss the simulation results for a numerical example to demonstrate the global convergence property of the proposed neural network.

Example. Consider the following (QP) problem and its dual (DQP):

(QP)	(DQP)
Min $x_1^2 + x_2^2 + x_1x_2 - 30x_1 - 30x_2,$	Max $\frac{35}{12}y_1 + \frac{35}{2}y_2 + 5y_3 + 5y_4 - x_1^2 - x_2^2 - x_1x_2,$
s.t. $\frac{5}{12}x_1 - x_2 + x_3 = \frac{35}{12},$	s.t. $\frac{5}{12}y_1 + \frac{5}{2}y_2 - y_3 - 2x_1 - x_2 \leq -30,$
$\frac{5}{2}x_1 + x_2 + x_4 = \frac{35}{2},$	$-y_1 + y_2 + y_4 - x_1 - 2x_2 \leq -30,$
$x_5 - x_1 = 5,$	$y_1 \leq 0,$
$x_2 + x_6 = 5,$	$y_2 \leq 0,$
$x_i \geq 0 \ (i = 1, 2, \dots, 6).$	$y_3 \leq 0,$
	$y_4 \leq 0.$

We have written a Matlab 6.5 code for solving (4) and executed the code on a Pentium IV. We have tested the model using all four possible initial points (in terms of feasibility and infeasibility) for the primal and dual problems. The numerical results obtained, as summarized in Table 1, show ultimate convergence to the optimal solutions $x^* = (5, 5, 5.833333, 0, 10, 0)^T$ for (QP) and $y^* = (0, -6, 0, -9)^T$ for (DQP). We will see that in all four possibilities of primal and dual feasible and infeasible starting points, the final result obtained is optimal with a high degree of accuracy.

Various trajectories from different initial points are shown in Figs. 3 and 4. Other aspects of the results in these figures are discussed below.

Case 1. The initial points of the primal problem are located within the feasible region (denoted by the area marked by the dashed line) and the initial point $y^0 = (0, -1, 0, -2)^T$ is given for the dual problem as indicated in Fig. 3. From this figure we see that trajectories always converge to the optimal solution, $x^* = (5, 5, 5.833333, 0, 10, 0)^T$.

Case 2. The initial points of the primal problem are outside the feasible region and the initial point $y^0 = (0, -1, 0, -2)^T$ is given for the dual problem as indicated in Fig. 4. Observe that in this case the trajectories also converge to the optimal solution, $x^* = (5, 5, 5.833333, 0, 10, 0)^T$.

Table 1
 Numerical results for (QP) and (DQP) problems using four different initial points (within feasible and infeasible regions) for the primal and dual problems

Initial point: primal variables	Initial point: dual variables	Primal (feasible or not feasible)	Dual (feasible or not feasible)	Optimal values for primal problem (x^*)	Optimal values for dual problem (y^*)	Primal		Dual	
						Optimal objective function value	Absolute error	Optimal objective function value	Absolute error
-5	0	Feasible	Feasible	4.999999999	-6.95673123e-011	-225.000000003	3.16106252e-010	-225.000000001	1.56887836e-010
-5	-12			5.000000000	-5.999999999				
0	24			5.833333334	2.73464745e-011				
35	-35			1.85089119e-033	-9.000000000				
0				9.999999999					
10				1.80498218e-034					
-5	0	Feasible	Not feasible	4.999999999	1.84478664e-011	-225.000000000	9.19442300e-011	-224.999999999	6.30109298e-011
-5	-1			5.000000000	-6.000000000				
0	0			5.833333333	-5.16284130e-012				
35	-2			2.10404664e-033	-8.999999999				
0				9.999999999					
10				2.34913816e-034					
3	0	Not feasible	Feasible	4.999999999	-8.92239124e-011	-225.000000004	4.56850557e-010	-225.000000001	1.94916083e-010
0	-12			5.000000000	-5.999999999				
7	24			5.833333334	3.56874380e-011				
5	-35			9.02491092e-035	-9.000000000				
-4				9.999999999					
1				1.80498218e-035					
3	0	Not feasible	Not feasible	4.999999999	-1.01132467e-011	-225.000000001	1.3304202184e-010	-225.000000000	1.2192913345e-011
0	-1			5.000000000	-5.999999999				
7	0			5.833333333	5.01038519e-012				
5	-2			9.41733623e-035	-8.999999999				
-4				9.999999999					
1				1.80498218e-035					

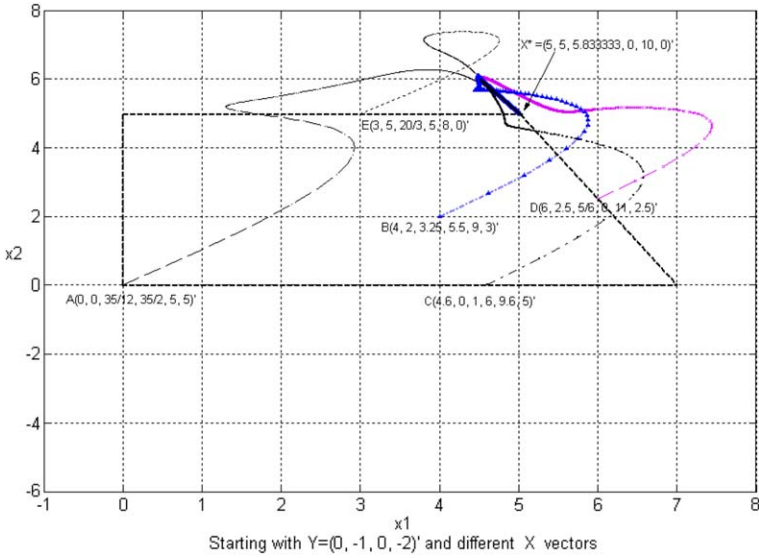


Fig. 3. QP trajectories with initial points inside the feasible region.

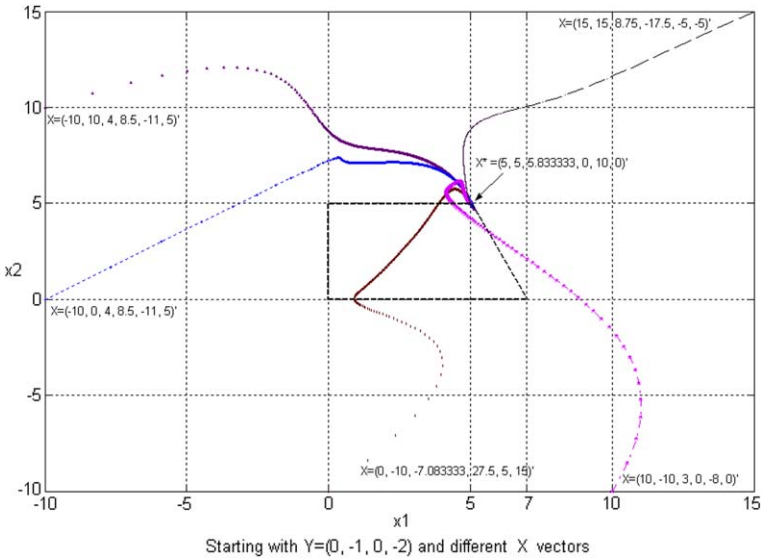


Fig. 4. QP trajectories with initial points outside the feasible region.

Thus the new neural network model always ultimately converges to the optimal solution, regardless of whether or not we choose the initial point within the feasible region. Hence, we may perceive the new model to be robust.

6. Concluding remarks

We have shown analytically and verified by simulation that our proposed neural network for solving the LP and QP problems is globally convergent. Our new neural network produces highly accurate solutions to the LP and QP problems and requires no analog multipliers for the variables. Hence, the proposed network, in several ways, improves over previously proposed models.

References

- [1] L.O. Chua, G.N. Lin, Nonlinear programming without computation, *IEEE Transactions on Circuits and Systems*, CAS 31 (2) (1984) 182–188.
- [2] G. Wilson, Quadratic programming analogs, *IEEE Transactions on Circuits and Systems*, CAS 33 (9) (1986) 907–911.
- [3] D.W. Tank, J.J. Hopfield, Simple neural optimization networks: an A/D converter, signal decision network, and linear programming circuit, *IEEE Transactions on Circuits and Systems*, CAS 33 (5) (1986) 533–541.
- [4] D.G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1989 (Chapter 12).
- [5] M.S. Bazaraa, C.M. Shetty, *Nonlinear Programming, Theory and Algorithms*, John Wiley and Sons, New York, 1990.
- [6] E.K.P. Chong, S. Hui, S.H. Zak, An analysis of a class of neural networks for solving linear programming problems, *IEEE Transactions on Automatic Control* 44 (11) (1999) 1995–2006.
- [7] A. Malek, H.G. Oskoei, Numerical solutions for constrained quadratic problems using high-performance neural networks, *Applied Mathematics and Computation*, in press, doi:10.1016/j.amc.2004.10.091.
- [8] A. Malek, A. Yari, Primal–dual solution for the linear programming problems using neural networks, *Applied Mathematics and Computation* 167 (1) (2004) 198–211.
- [9] Y. Leung, K. Chen, X. Gao, A high-performance feedback neural network for solving convex nonlinear programming problems, *IEEE Transactions on Neural Networks* 14 (6) (2003) 1469–1477.
- [10] Y. Leung, K. Chen, Y. Jiao, X. Gao, K.S. Leung, A new gradient-based neural network for solving linear and quadratic programming problems, *IEEE Transactions on Neural Networks* 12 (5) (2001) 1074–1083.
- [11] M.P. Kennedy, L.O. Chua, Neural network for nonlinear programming, *IEEE Transactions on Circuits and Systems*, CAS 35 (5) (1988) 554–562.
- [12] C.Y. Maa, M.A. Shanblatt, Linear and quadratic programming neural network analysis, *IEEE Transactions on Neural Networks* 3 (4) (1992) 580–594.
- [13] A. Rodriguez-Vazquez, R. Dominguez-Castro, A. Rueda, J.L. Huertas, E. Sanchez-Sinencio, Nonlinear switched-capacitor neural networks for optimization problems, *IEEE Transactions on Circuits and Systems* 37 (3) (1990) 384–398.
- [14] S.H. Zak, V. Upatising, S. Hui, Solving linear programming problems with neural networks: a comparative study, *IEEE Transactions on Neural Networks* 6 (1) (1995) 96–104.
- [15] X. Wu, Y. Xia, J. Li, W. Chen, A high performance neural network for solving linear and quadratic programming problems, *IEEE Transactions on Neural Networks* 7 (3) (1996) 643–651.
- [16] Y. Xia, A new neural network for solving linear programming problems and its application, *IEEE Transactions on Neural Networks* 7 (2) (1996) 525–529.

- [17] Y. Xia, A new neural network for solving linear and quadratic programming problems, *IEEE Transactions on Neural Networks* 7 (6) (1996) 1544–1547.
- [18] S.L. Ross, *Introduction to Ordinary Differential Equations*, fourth ed., Wiley, New York, 1989.