Available online at http://ijim.srbiau.ac.ir/
Int. J. Industrial Mathematics (ISSN 2008-5621)
Vol. 10, No. 4, 2018 Article ID IJIM-01083, 9 pages
Research Article

A Recurrent Neural Network for Solving Strictly Convex Quadratic Programming Problems

A. Ghomashi *†, M. Abbasi ‡

Abstract

In this paper, we present an improved neural network to solve strictly convex quadratic programming(QP) problem. The proposed model includes a set of differential equations such that their equilibrium points correspond to optimality condition of convex (QP) problem and has a lower structure complexity respect to the other existing neural network model for solving such problems. In theoretical aspect, stability and global convergence of the proposed neural network is proved. The validity and transient behavior of the proposed neural network are demonstrated by using four numerical examples.

Keywords : Dynamical system; Strictly convex quadratic programming; Stability; Global convergence; Recurrent neural networks

1 Introduction

One promising approach to solving the optimization problems with hight dimension and dense structure in real time is to employ artificial neural networks based on circuit implementation [24]. Neural networks are computing systems composed of a number of highly interconnected simple information processing units, and thus can usually solve optimization problems in execution times at the orders of magnitude much faster than most popular optimization algorithms for general-purpose digital [24]. A neural network with a good computational performance should satisfy threefold. First, the global convergence of the neural networks with an arbitrarily given initial state should be guaranteed. Second, the network design preferably contains no variable parameter. Third, the equilibrium points of the network should correspond to the exact or approximate solution [17]. Solving optimization problems using recurrent neural networks has fascinated much attention since seminal work of Tank and Hopfield [11]. Many neural network for constrained optimization problems has been developed during the past two decades,e.g.see [2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 15, 17, 18, 19, 20, 21, 22, 24, 26, 28, 30] and references therein. Kennedy and Chua [4] presented a neural network for solving the strictly convex quadratic programming. Proposed model by kennedy and chua contain a finite penalty parameter, thus their model converges to an approximate optimal solution. Lately, many researchers successively proposed a number of primal-dual neural networks [8, 9, 28] and projection neural networks for solving linear and quadratic programming problems [2, 6, 15, 18, 20, 23, 24, 26]. Moreover, a number of neural networks models proposed for solving a

*Corresponding author. a_ghomashi_l@iauksh.com, Tel:+98(918)7260024.
†Department of Mathematics, Kermanshah Branch, Islamic Azad University, Kermanshah, Iran.
‡Department of Mathematics, Kermanshah Branch, Islamic Azad University, Kermanshah, Iran.

special forms of quadratic programming problems such as L_1-norm estimation and L_2-norm estimation problems [14, 16, 25]. These neural networks were proved to be globally convergent to the exact solutions.
We are concerned with the following QP problem:

$$\begin{aligned} &\text{Minimize } c^T x + \tfrac{1}{2} x^T Q x \\ &\text{subject to} \\ &\qquad \bar{l} \le Ax \le \bar{h} \\ &\qquad l \le x \le h \end{aligned} \tag{1.1}$$

where $c, x \in R^n, A \in R^{m\times n}, \bar{l}$ and $\bar{h} \in R^m$, $Rank(A) = m$ and $Q \in R^{n\times n}$ is symmetric and positive definite matrix.
Problem (1.1) is a general form of quadratic programming problem. In the most of existing neural network models for solving (1.1), constraint $\bar{l} \le Ax \le \bar{h}$ divided two part, $\bar{l} \le Ax$ and $Ax \le \bar{h}$, thus dimension of problem increases. First time, Xia et al. studied [22] a neural network for solving (1.1) without dividing $\bar{l} \le Ax \le \bar{h}$ into two part. Proposed model in [22] motivates us to propose a neural network model with one layer structure for solving (1.1) with lower model complexity respect to proposed model in [22]. The proposed neural network is shown to to be globally convergent to unique exact solution of (1.1)within a finite time. Simulation results show that the proposed neural network is effective for solving strictly convex quadratic programming problems. This paper is divided into six sections. In next section preliminary information is introduced to facilitate later discussions. In section III, first we present that problem (1.1) is equivalent with solving a piecewise equation and then introduce a neural network model for solving this piecewise equation. In section IV, we analyze stability condition and global convergence. In section V, illustrative examples are discussed. Section VI gives the conclusion of this paper.

2 Preliminaries

Definition 2.1 *Let* $X = \{x \in R^m | l_i \le x_i \le u_i, \forall i \in N \subseteq L\}$ *where* $L = \{1, 2, ..., m\}$, $P_X : R^m \to X$ *is a projection operator to set* X *defined by*

$$P_X(x) = \underset{y \in X}{arg\ min}\ \|x - y\|$$

Where $\|.\|$ *denotes the* l_2*-norm of* R^m *[24]. Since* X *is a box set,* $P_X(x)$ *can be presented by* $P_X(x) = [P_X(x_1), ..., P_X(x_m)]^T$, *where for* $i \in L - N, P_X(x_i) = x_i$ *and for* $i \in N$

$$P_X(x_i) = \begin{cases} l_i, & x_i < l_i \\ x_i & l_i \le x_i \le u_i \\ u_i & x_i > u_i \end{cases} \tag{2.2}$$

Lemma 2.1 *[5] For all* $y \in R^m$ *and all* $x \in X \subseteq R^m$

$$(y - P_X(y))^T (P_X(y) - x) \ge 0$$

and for $x, y \in R^m$

$$\|P_X(x) - P_X(y)\| \le \|x - y\|$$

Definition 2.2 *The finite-dimensional variational inequality problem* $VI(F, K)$ *is to determine a vector* $x^* \in K \subseteq R^n$, *such that*

$$F(x^*)^T (x - x^*) \ge 0, \qquad \forall x \in K$$

where F *is a given continuous function from* K *to* R^m, K *is a given closed convex set [5].*

Theorem 2.1 *[5] Assume that* K *is closed and convex. Then* $x^* \in K$ *is a solution of the variational inequality problem* $VI(F, K)$ *if and only if for any* $\gamma > 0$

$$P_K(x^* - \gamma F(x^*)) = x^*$$

Let $f : R^n \to R^n$, now we introduce some basic properties of the following differential equation:

$$\dot{x}(t) = f(x(t)), \quad x(t_0) \in R^n \tag{2.3}$$

Theorem 2.2 *[29] Assume that* f *in (2.3) is a continuous mapping, then for arbitrary* $t_0 \ge 0$ *and* $x_0 \in R^n$ *there exists a local solution* $x(t)$ *to (2.3) where* $t \in [t_0, \tau]$ *for some* $\tau > t_0$. *Furthermore if* f *is locally Lipschitzian continuous at* x_0 *then the solution is unique, and if* f *is Lipschitzian continuous in* R^n *then* τ *can be extended to* $+\infty$.

Theorem 2.3 *[29] Let* x^* *is an equilibrium point of (2.3) and* $X \subset R^n$ *be an open neighborhood of* x^*, *if* $V : R^n \to R$ *is a continuously differentiable function over* X *and* V *satisfies in the following conditions:*

- $V(x^*) = 0, \quad \frac{dV(x^*)}{dt} = 0$

- $\frac{dV(x(t))}{dt} \leq 0, \quad V(x) > 0, \ \forall x \in X - \{x^*\}$
- $\|x\| \to \infty \Rightarrow \|V(x)\| \to \infty$

then x^* is a lyapunov stable equilibrium and the solution always exist globally,if

$$\frac{dV(x(t))}{dt} < 0 \quad \forall x \in X - \{x^*\}$$

then x^* is a globally asymptotically stable equilibrium.

Theorem 2.4 *[1] Let S be a nonempty open convex set in R^n, and letf $: S \to R$ be differentiable on S then f is strictly convex if and only if, for each distinct $x_l, x_2 \in S$, we have*

$$(\nabla f(x_2) - \nabla f(x_1))^T (x_2 - x_1) > 0$$

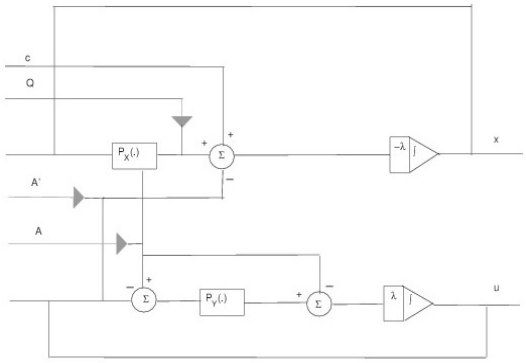

Figure 1: Architecture of the recurrent neural network (3.7).

3 Neural network model

Throughout this paper, we assume that the feasible set of problem(1.1)is nonempty.
The Karush-Kuhn-Tucker conditions of (1.1) has the following form [1]:

$$Qx + c + A^T u = 0, \tag{3.4}$$

$$\begin{cases} (Ax)_i = \bar{l}_i, & u_i > 0 \\ \bar{l}_i \leq (Ax)_i \leq \bar{h}_i & u_i = 0 \\ (Ax)_i = \bar{h}_i, & u_i < 0 \end{cases} \tag{3.5}$$

where $x \in X = \{x \in R^n | l \leq x \leq h\}$ and $u \in R^m$.

Lemma 3.1 *[21] $\bar{x} \in X$ is an optimal solution of (1.1) if and only if there exist $\bar{u}$ such that $(\bar{x}, \bar{u})$ satisfies in (3.4) and (3.5).*

$\bar{x}$ is called a KKT point of (1.1) and $\bar{u}$ is called the lagrangian multiplier vector corresponding to $\bar{x}$.
Now, let $x(.)$ and $u(.)$ be some time dependent variables. In order to use a neural network method to solve (1.1), a neural network system have to be constructed and make the steady points of neural network system to satisfy the KKT conditions (3.4) and (3.5). By definition 2.1, we can rewrite (3.4) and (3.5) the following form:

$$\begin{cases} QP_X(x) + c - A^T u = 0 \\ P_Y(AP_X(x) - u) = AP_X(x) \end{cases} \tag{3.6}$$

where $Y = \{y \in R^m | \bar{l} \leq y \leq \bar{h}\}$, $X = \{x \in R^n | l \leq x \leq h\}$, $u \in R^m$ and $x \in R^n$.
Using (3.6), we propose a new neural network for solving (1.1) as follows:

$$\begin{aligned} \frac{dx}{dt} &= -\lambda(QP_X(x) + c - A^T u) \\ \frac{du}{dt} &= -\lambda(AP_X(x) - P_Y(AP_X(x) - u)) \end{aligned} \tag{3.7}$$

where $\lambda > 0$ is scalar. The architecture of the neural network described in (3.7)is depicted in Figure 1. The system described by Eq.(3.7) can be applied for solving (1.1) with positive definite matrix Q and can be easily realized by a recurrent neural network with a one-layer structure.The proposed neural network can be implemented by using a simple hardware only without analog multipliers for the variables or the penalty parameter. The operator $P_Y$andP_X may be implemented by using a piecewise activation function. The model contains some amplifiers($P_Y$andP_X), integrator, summations, multipliers and interconnections. Among them, the number of amplifiers, integrator and interconnections is important in determining the structural complexity of neural network model.
For comparison purpose, we list the numbers of amplifiers, integrators and interconnections of proposed neural network in (3.7)and proposed neural network in [22] in Table 1. we have the following observations: 1) the numbers of amplifiers and integrator are same in both models; 2) the proposed model requires fewest interconnections respect to model in [22]. These observations then lead to the conclusion that the proposed model (3.7) is simplest in structure. In what follows, we introduce some basic properties of (3.7).

Theorem 3.1 *$\bar{x} = P_X(x^*)$ is optimal solution*

Table 1: Comparisons of proposed model and model in [22] for solving (1).

Neural network model	Number of amplifiers	number of integrator	number of interconnections
proposed model	$m+n$	$m+n$	$n^2+2mn+3n+5m$
model in [22]	$m+n$	$m+n$	$(m+n+6)(m+n)$

of (1.1) where $\begin{bmatrix} x^* \\ u^* \end{bmatrix}$ *denotes the equilibrium point of (3.7).*

Proof. *Using lemma 3.1, proof is complete.*

Corollary 3.1 *Right hand side of (3.7) is Lipschitz continuous function.*

Proof. Let the right hand side of (3.7) be denoted by $L(w)$, where $w = \begin{bmatrix} x \\ u \end{bmatrix} \in R^{m+n}$, by lemma 2.1, for any $\hat{w} = \begin{bmatrix} \hat{x} \\ \hat{u} \end{bmatrix} \in R^{m+n}$, and $\bar{w} = \begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix} \in R^{m+n}$, we have:

$$\|L(\hat{w}) - L(\bar{w})\| = \left\| \lambda \begin{pmatrix} Q(P_X(\bar{x}) - P_X(\hat{x})) + A^T(\hat{u} - \bar{u}) \\ P_Y(AP_X(\hat{x}) - \hat{u}) - P_Y(AP_X(\bar{x}) \\ -\bar{u}) + A(P_X(\bar{x}) - P_X(\hat{x})) \end{pmatrix} \right\|$$

$$\leq \left\| \lambda \begin{pmatrix} Q(\bar{x} - \hat{x}) + A^T(\hat{u} - \bar{u}) \\ A(P_X(\hat{x}) - P_X(\bar{x})) \\ \quad + \bar{u} - \hat{u} - A(\hat{x} - \bar{x}) \end{pmatrix} \right\|$$

$$\leq \left\| \lambda \begin{pmatrix} Q(\bar{x} - \hat{x}) + A^T(\hat{u} - \bar{u}) \\ \bar{u} - \hat{u} \end{pmatrix} \right\|$$

$$\leq \left\| \lambda \begin{pmatrix} -Q & A^T \\ 0 & -I \end{pmatrix} \begin{pmatrix} \hat{x} - \bar{x} \\ \hat{u} - \bar{u} \end{pmatrix} \right\|$$

$$\leq \left\| \lambda \begin{pmatrix} -Q & A^T \\ 0 & -I \end{pmatrix} \right\| \left\| \begin{pmatrix} \hat{x} \\ \hat{u} \end{pmatrix} - \begin{pmatrix} \bar{x} \\ \bar{u} \end{pmatrix} \right\|$$

Which gives the desired results.

Lemma 3.2 *For each initial point* $\begin{bmatrix} x(t_0) \\ u(t_0) \end{bmatrix} \in R^{m+n}$, *there exist a unique continuous solution* $\begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in R^{m+n} (t \in [t_0, \tau))$ *for (3.7) and the equilibrium of neural network in (3.7) correspond to unique optimal solution of (1.1).*

Proof. Theorem 2.2 and Corollary 3.1 yield a unique continuous solution $\begin{bmatrix} x(t) \\ u(t) \end{bmatrix}$ over $[t_0, \tau)$ exist for (3.7). Since the feasible set of problem(1.1) is nonempty and Q is a positive definite matrix so there exist a unique optimal solution for (1.1) [1], then using theorem 3.1 and (3.7), proof is complete.

4 Convergence Analysis

In this section, we prove globally asymptotically convergent of (3.7).
The neural network in (3.7) is said to be stable in the sense of Lyapunov and globally convergent, globally asymptotically stable, if the corresponding dynamic system is so [24].

Theorem 4.1 *The proposed neural network in (3.7) is stable in the sense of Lyapunov and is globally asymptotically convergent to the unique solution of (1.1) if Q is positive definite. Moreover, the convergence rate of the neural network in (3.7) increase as λ increases.*

Proof. By Lemma 3.2, we know that for each initial point $\begin{bmatrix} x(t_0) \\ u(t_0) \end{bmatrix} \in R^{m+n}$, there exist a unique continuous solution $\begin{bmatrix} x(t) \\ u(t) \end{bmatrix}$ for (3.7). Let $\begin{bmatrix} x^* \\ u^* \end{bmatrix}$ be equilibria point of (3.7), define a lyapunov function below:

$$V(x(t), u(t)) = \tfrac{1}{2} \| x(t) - P_X(x^*) \|^2 + \tfrac{1}{2} \| u(t) - u^* \|^2, \qquad \forall t \geq t_0,$$

Let $x = x(t)$ and $u = u(t)$, then time derivative of V along the trajectory of (3.7) as follows:

$$\frac{d}{dt} V(x, u) = \frac{dV}{dx}\frac{dx}{dt} + \frac{dV}{du}\frac{du}{dt} \tag{4.8}$$

We have

$$\frac{dV}{dx}\frac{dx}{dt} = -\lambda(QP_X(x) + c - A^T u)^T (x - P_X(x^*))$$

Define $g(x) = c^T x + \frac{1}{2} x^T Q x - u^T A x$ where u is fix scaler. Since Q is positive definite so $g(x)$ is strictly convex. since $\begin{bmatrix} x^* \\ u^* \end{bmatrix}$ be equilibrium point

of (3.7) so $\nabla g(P_X(x^*)) = QP_X(x^*) + c - A^T u^* = 0$, using theorem 2.4 we have

$$(\nabla g(x) - \nabla g(P_X(x^*))^T (x - P_X(x^*)) > 0, \quad x \neq P_X(x^*)$$

So

$$\frac{dV}{dx}\frac{dx}{dt} = -\lambda(QP_X(x) + c - A^T u)^T (x - P_X(x^*)) < 0, \quad x \neq P_X(x^*) \tag{4.9}$$

On the other hand by lemma 2.1 we have:

$$(v - P_Y(v))^T (P_Y(v) - y) \geq 0, \ v \in R^m, y \in Y$$

Let $v = AP_X(x) - u$ and $y = AP_X(x^*)$ then

$$(AP_X(x) - u - P_Y(AP_X(x) - u))^T (P_Y(AP_X(x) - u) - AP_X(x^*)) \geq 0, \quad \forall x \in R^n, \ \forall u \in R^m \tag{4.10}$$

By using definition 2.2 and theorem 2.1 we have:

$$(u^*)^T (y - AP_X(x^*)) \geq 0, \qquad \forall y \in Y$$

Let $y = P_Y(AP_X(x) - u)$ so

$$(u^*)^T (P_Y(AP_X(x) - u) - P_X(x^*)) \geq 0, \quad \forall x \in R^n, \ \forall u \in R^m \tag{4.11}$$

Sum of (4.10) and (4.11) yields:

$$(AP_X(x) - u + u^* - P_Y(AP_X(x) - u))^T (P_Y(AP_X(x) - u) - AP_X(x^*)) \geq 0, \quad \forall x \in R^n, \ \forall u \in R^m$$

Then

$$(u - u^*)^T (P_Y(AP_X(x) - I(u)) - AP_X(x)) \leq -\|(P_Y(AP_X(x) - I(u)) - AP_X(x))\|^2 - (u - u^*)^T (AP_X(x) - AP_X(x^*)) \tag{4.12}$$

By using (3.6) we have

$$P_X(x) = Q^{-1}A^T u - Q^{-1}c \qquad P_X(x^*) = Q^{-1}A^T u^* - Q^{-1}c \tag{4.13}$$

Substitution (4.13) into (4.12) yields

$$(u - u^*)^T (P_Y(AP_X(x) - I(u)) - AP_X(x)) \leq -\|(P_Y(AP_X(x) - I(u)) - AP_X(x))\|^2 - (u - u^*)^T AQ^{-1}A^T (u - u^*) \leq 0 \tag{4.14}$$

Then

$$\frac{dV}{du}\frac{du}{dt} = \lambda(u - u^*)^T (P_Y(AP_X(x) - I(u)) - AP_X(x)) \leq 0 \tag{4.15}$$

By using(4.8),(4.9) and (4.15) we have

$$\frac{d}{dt}V(x,u) = \frac{dV}{dx}\frac{dx}{dt} + \frac{dV}{du}\frac{du}{dt} < 0, \quad \forall (x,u) \neq (P_X(x^*), u^*)$$

So we have

$$\{(x(t), u(t)) | t_0 \leq t < \tau\} \subset P_0 = \{(x,u) \in R^{m+n} | V(x,u) \leq V(x(t_0), u(t_0))\}$$

On the other hand we have:

$$V(x,u) \geq \tfrac{1}{2} \| x - P_X(x^*) \|^2, \qquad V(x,u) \geq \tfrac{1}{2} \| u - u^* \|^2$$

Since P_0 is bounded and $\{(x(t), u(t)) | t_0 \leq t < \tau\} \subset P_0$, $(x(t), u(t))$ is bounded and thus $\tau = \infty$. Moreover

$$\frac{dV(x,u)}{dt} = 0 \Leftrightarrow \begin{cases} QP_X(x) + c - A^T u = 0 \\ P_Y(AP_X(x) - u) - AP_X(x) = 0 \end{cases} \Leftrightarrow \begin{cases} \frac{dx}{dt} = 0 \\ \frac{du}{dt} = 0 \end{cases}$$

So by applying the theorem 2.3, we get result that the proposed neural network is globally asymptotically convergent to the unique solution of (1.1).
Since $\frac{dV}{dt} < 0$ then we can result that as λ increases, the convergence rate of the neural network in (3.7) increases. This proof is completed. □

5 Illustrative examples

In this section, we demonstrate the effectiveness and performance of the proposed neural network model with four illustrative examples. The ordinary differential equation solver engaged in ode23 in matlab 2011.

Example 5.1 *Consider the following quadratic programming problem [22]:*

$$\begin{aligned} \text{Minimize} \quad & x_1^2 + x_2^2 + x_1 x_2 - 30x_1 - 30x_2 \\ \text{subject to} \quad & \tfrac{5}{12}x_1 - x_2 \leq \tfrac{35}{12} \\ & \tfrac{5}{2}x_1 + x_2 \leq \tfrac{35}{2} \\ & -5 \leq x_1 \leq 5 \\ & -5 \leq x_2 \leq 5 \end{aligned}$$

Optimal solution of the above problem is $x^* = (5,5)$. We use the neural network in (3.7) to solve this problem. All simulation results show that the

neural network in (3.7) is globally asymptotically stable to x^*. Figure 2 shows the performance of the neural network in (3.7) with a random initial point and four λ.

Example 5.2 *Consider the following quadratic programming problem [22]:*

$$\begin{aligned} \text{Minimize} \quad & x_1^2 + x_2^2 + 5x_3^2 + x_1x_2 \\ & + x_1x_3 - 4x_1 - 3x_2 - 2x_3 \\ \text{subject to} \quad & x_1 + x_2 + 2x_3 \leq 3 \\ & 3x_1 - 9x_2 + 9x_3 = 1 \\ & 0 \leq x_1,\ x_2,\ x_3 \leq \frac{4}{3} \end{aligned}$$

This problem has a unique optimal solution $x^* = (\frac{4}{3}, \frac{7}{9}, \frac{4}{9})$. We use the proposed neural network in (3.7) to solve Example 5.2. Figure 3 and Figure 4 display the convergence behavior proposed model in example 5.2.

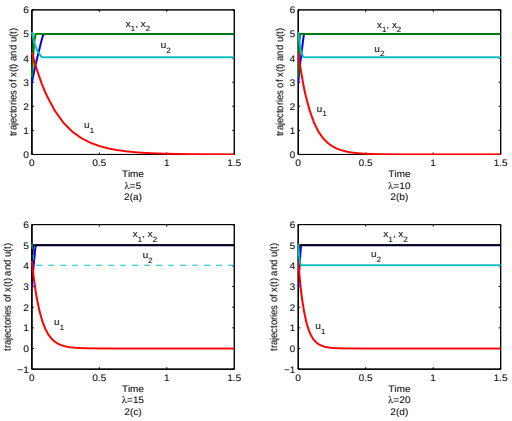

Figure 2: Transient behavior of the neural network in (3.7) in terms of trajectories in Example 5.1.

Example 5.3 *Consider the following quadratic programming problem:*

$$\begin{aligned} \text{Minimize} \quad & 3x_1^2 + 3x_2^2 + 4x_3^2 + 5x_4^2 + 3x_1x_2 \\ & + 5x_1x_3 + x_2x_4 - 11x_1 - 5x_4 \\ \text{subject to} \quad & 3x_1 - 3x_2 - 2x_3 + 4x_4 = 0 \\ & 4x_1 + x_2 - x_3 - 24x_4 = 0 \\ & -x_1 + x_2 \leq -1 \\ & -2 \leq 3x_1 + x_3 \leq 4 \end{aligned}$$

This problem has an optimal solution $x^* = (0.5, -0.5, 1.5, 0)^T$. We use the proposed neural network in (3.7) to solve Example 5.3. Figure 5.a illustrates the convergence behavior of the l_2 norm error $\|x(t) - x^*\|$ based on the neural network in (3.7). Figure 5.b displays the trajectories of the state trajectories $x(t)$ started from 10 random initial points.

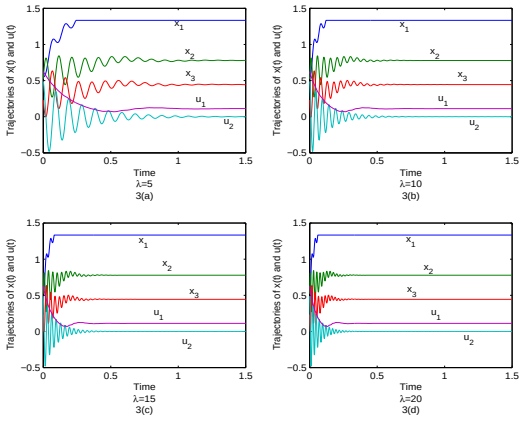

Figure 3: Transient behavior of the neural network in (3.7) in terms four λ and one random initial point to solve Example 5.2.

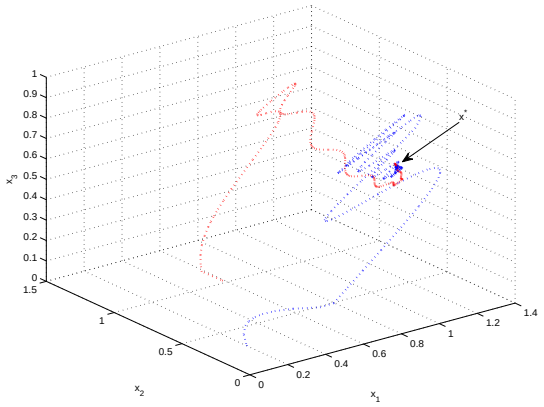

Figure 4: Transient behavior of the neural network in (3.7) in terms of two initial points to solve Example 5.2.

Example 5.4 *Consider the quadratic programming problem (1.1) [22] where*

$$Q = \begin{pmatrix} 2 & 1 & 0 & \dots & 0 \\ 1 & 2 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 2 & 1 \\ 0 & \dots & 0 & 1 & 2 \end{pmatrix}$$

$c = [-1, 4, -1, 1, 0, 0, 1, 0, 1, 0]^T$,
$l = [1, -1, 0]^T$, $u = [7, 5, 1]^T$

Table 2: Results of (3.7) and proposed model in [22] for Example 5.4.

Model	iteration 1		iteration 2		iteration 3		iteration 4	
	CPU	Error	CPU	Error	CPU	Error	CPU	Error
model(7)	0.2188	1×10^{-5}	0.2344	8×10^{-6}	0.2188	7×10^{-6}	0.2344	8×10^{-6}
model in [22]	1	3×10^{-5}	0.9844	1×10^{-5}	0.8906	3×10^{-5}	0.8750	2×10^{-5}

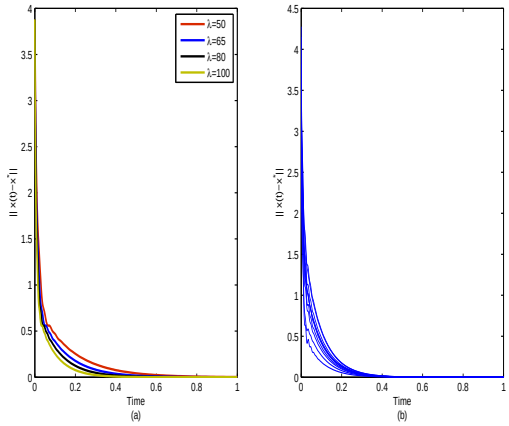

Figure 5: Convergence behavior of the proposed model in terms of the norm error $\|x(t) - x^*\|$ in Example 5.3. (a) with one random initial point and four λ. (b) with 10 random initial points and $\lambda = 200$.

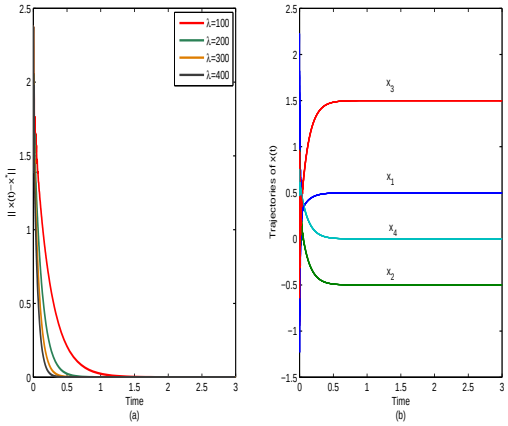

Figure 6: (a) Convergence behavior of the proposed model in terms of the norm error $\|x(t)-x^*\|$ in Example 5.4 with one random initial points and four λ. (b) Transient behavior of the proposed model in Example 5.4 in terms of one random initial point and $\lambda = 100$

and

$$A = \begin{pmatrix} 1 & -1 & -1 & 1 & -1 & 1 & 0 & 1 & 1 & -1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ -1 & 1 & -1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Optimal solution of the above problem is $x^* = (0.5, -1.5, 0, 1, -1.5, 2, -1, 0.5, 0, 0)$. We use the neural network in (3.7) to solve this problem. Figure 6 shows the transient behavior of the proposed model in Example 5.4.

For a comparison, we compute this example using the proposed neural network in (3.7) and proposed model in [22] in four iteration. The computational results are listed in Table 2. From Table 2, we see that the proposed neural network not only gives a better solution, but also has faster convergence rate than proposed model in [22].

6 Conclusion

In this paper, a recurrent neural network introduced for solving strictly convex quadratic programming problem so it can solve a broad class of the constrained optimization problems. The proposed model has a lower structure complexity respect to existing models to solve such problem. It is shown here that the proposed neural network is stable in the sense of Lyapunov and globally asymptotically convergent to the optimal solution. Numerical examples are provided to show the performance of the proposed neural network.

References

[1] S. Boyd, L. Vandenberghe, Convex optimization, *Cambridge university press* (2004).

[2] X. Hu, Applications of the general projection neural network in solving extended linear-quadratic programming problems with linear constraints, *Neurocomputing* 72(2004) 1131-1137.

[3] X. Hu, B. Zhang, A New Recurrent Neural Network for Solving Convex Quadratic Programming Problems With an Application tothe-Winners-Take-All Problem, *Neu-*

ral Networks, IEEE Transactions on 20 (2009) 654-664.

[4] M. P. Kennedy, L. O. Chua, Neural networks for nonlinear programming. Circuits and Systems, *IEEE Transactions on* 35 (1988) 554-562.

[5] D. Kinderlehrer, G. Stampacchia, An introduction to variational inequalities and their applications (Vol. 31), *Siam* (2000).

[6] Q. Liu, J. Wang, A one-layer projection neural network for nonsmooth optimization subject to linear equalities and bound constraints. Neural Networks and Learning Systems, *IEEE Transactions on* 24 (2013) 812-824.

[7] C. Y. Maa, M. A. Shanblatt, Linear and quadratic programming neural network analysis, Neural Networks, *IEEE Transactions on* 3 (1992) 580-594.

[8] A. Nazemi, A neural network model for solving convex quadratic programming problems with some applications, *Engineering Applications of Artificial Intelligenc* 32 (2014) 54-62.

[9] A. Nazemi, M. Nazemi, A Gradient-Based Neural Network Method for Solving Strictly Convex Quadratic Programming Problems, *Cognitive Computation* 5 (2014)1-12.

[10] A. Rodriguez-Vazquez, R. Dominguez-Castro, A. Rueda, A., J. L. Huertas, E. Sanchez-Sinencio, Nonlinear switched capacitor neural networks for optimization problems. Circuits and Systems, *IEEE Transactions on* 37 (1990) 384-398.

[11] D. Tank, J. J. Hopfield, Simple'neural'optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. Circuits and Systems, *IEEE Transactions on* 33 (1986) 533-541.

[12] X. Y. Wu, Y. S. Xia, J. Li, W. K. Chen, A high-performance neural network for solving linear and quadratic programming problems. Neural Networks, *IEEE Transactions on* 7(1986) 643-651.

[13] Y. Xia, A new neural network for solving linear and quadratic programming problems, Neural Networks, *IEEE Transactions on* 7(1996) 1544-1548.

[14] Y. Xia, A Compact Cooperative Recurrent Neural Network for Computing General Constrained Norm Estimators, Signal Processing, *IEEE Transactions on* 57 (2009) 3693-3697.

[15] Y. S. Xia, J. Wang, On the stability of globally projected dynamical systems, *Journal of Optimization Theory and Applications* 106 (2000) 129-150.

[16] Y. Xia, H. Leung, A Fast Learning Algorithm for Blind Data Fusion Using a Novel-Norm Estimation, *Sensors Journal, IEEE* 14 (2014) 666-672.

[17] Y. Xia, J. Wang, A general methodology for designing globally convergent optimization neural networks, *Neural Networks, IEEE Transactions on* 9 (1998) 1331-1343.

[18] Y. Xia, J. Wang, A recurrent neural network for solving linear projection equations, *Neural Networks* 13 (2000) 337-350.

[19] Y. Xia, J. Wang, Global exponential stability of recurrent neural networks for solving optimization and related problems, *Neural Networks, IEEE Transactions on* 11 (2000) 1017-1022.

[20] Y. Xia, J. Wang, A general projection neural network for solving monotone variational inequalities and related optimization problems, *Neural Networks, IEEE Transactions on* 15 (2004) 318-328.

[21] Y. Xia, J. Wang, A recurrent neural network for solving nonlinear convex programs subject to linear constraints, *Neural Networks, IEEE Transactions on* 16 (2005) 379-386.

[22] Y. Xia, G. Feng, J. Wang, A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equations, *Neural Networks* 17 (2004) 1003-1015.

[23] Y. Xia, G. Feng, J. Wang, A novel recurrent neural network for solving nonlinear

optimization problems with inequality constraints, *Neural Networks, IEEE Transactions on* 19 (2008) 1340-1353.

[24] Y. Xia, H. Leung, J. Wang, A projection neural network and its application to constrained optimization problems.Circuits and Systems I: Fundamental Theory and Applications, *IEEE Transactions on* 49 (2002) 447-458.

[25] Y. Xia, C. Sun, W. X. Zheng, Discrete-time neural network for fast solving large linear estimation problems and its application to image restoration, *Neural Networks and Learning Systems, IEEE Transactions on* 23 (2012) 812-820.

[26] X. Xue, W. Bian ,A project neural network for solving degenerate convex quadratic program, *Neurocomputing* 70 (2007) 59-66.

[27] Y. Yan, A New Nonlinear Neural Network for Solving QP Problems, In Advances in Neural Networks ISNN 2014 (pp. 347-357), *Springer International Publishing* (2014).

[28] Y. Yang, J. Cao, A feedback neural network for solving convex constraint optimization problems, *Appl Math Comput.* 201 (2008) 50-74.

[29] J. Zabczyk, Mathematical control theory: an introduction, *Springer* (2009).

[30] J. Zhang, L. Zhang, An augmented lagrangian method for a class of inverse quadratic programming problems, *Appl Math Optim.* 61 (2010) 57-83.

Abbas Ghomashi received the MS degrees in Applied Mathematics from Tarbiat Moallem University of Sabzevar, Sabzevar, Iran in 2004 and PhD degree from Science and Research Branch, Islamic Azad University, Tehran, Iran in 2014. Now, he is an assistant professor in Department of Mathematics, Kermanshah Branch, Islamic Azad University, Iran. His research interests include Neural Networks, Linear and nonlinear programming, Data Envelopment Analysis and Interior Point Methods.

Masomeh Abbasi received the MS degrees in Applied Mathematics from Tarbiat Moallem University of Sabzevar, Sabzevar, Iran in 2005 and PhD degree from Science and Research Branch, Islamic Azad University, Tehran, Iran in 2014. Now, she is an assistant professor in Department of Mathematics, Kermanshah Branch, Islamic Azad University, Iran. Her research interests include Linear and nonlinear programming, Data Envelopment Analysis and Interior Point Methods.