A New One-Layer Neural Network for Linear and Quadratic Programming

Xingbao Gao and Li-Zhi Liao

***Abstract*—In this paper, we present a new neural network for solving linear and quadratic programming problems in real time by introducing some new vectors. The proposed neural network is stable in the sense of Lyapunov and can converge to an exact optimal solution of the original problem when the objective function is convex on the set defined by equality constraints. Compared with existing one-layer neural networks for quadratic programming problems, the proposed neural network has the least neurons and requires weak stability conditions. The validity and transient behavior of the proposed neural network are demonstrated by some simulation results.**

***Index Terms*—Convergence, linear and quadratic programming, neural network, stability.**

I. Introduction

CONSIDER the following quadratic programming problem:

$$\begin{cases} \min & f(x) = \frac{1}{2}x^T Ax + a^T x \\ \text{s.t.} & Bx \in \mathcal{B}, \qquad Dx = d, \qquad x \in \mathcal{X} \end{cases} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)^T \in R^n$ is the unknown variable, $A = A^T \in R^{n\times n}$, $a \in R^n$, $B \in R^{m\times n}$, $D \in R^{p\times n}$ ($\text{rank}(D) = p, 0 \le p < n$), $d \in R^p$, $\mathcal{X} = \{x \in R^n \mid \underline{l}_i \le x_i \le \overline{l}_i,\ i = 1, 2, \ldots, n\}$, $\mathcal{B} = \{\lambda \in R^m \mid \underline{h}_j \le \lambda_j \le \overline{h}_j,\ j = 1, 2, \ldots, m\}$, and some $-\underline{l}_i$ (or $-\underline{h}_j$, $\overline{l}_i$, and $\overline{h}_j$) could be $+\infty$. Clearly, (1) includes equality and inequality constraints, and linear programming problems ($A = 0$).

Linear and quadratic programming problems arise in a wide variety of scientific and engineering fields (see [1], [2], [5], [6], [17], and [24]) including regression analysis, function approximation, signal processing, image restoration, parameter estimation, filter design, robot control, etc.; a real-time solution is often desired. However, traditional numerical methods (see [1], [2], [5], [17], [18], [29], [32], [35], and the references therein) might not be efficient for digital computers since the computing time required for a solution is greatly dependent on the dimension and the structure of the problem, and the complexity of the algorithm used. One promising approach to handle these optimization problems with high dimension and dense structure is to employ artificial-neural-network-based circuit implementation. Because of the dynamic nature for optimization and the potential of electronic implementation, neural networks can be implemented physically by designated hardware such as application-specific integrated circuits where the optimization procedure is truly parallel and distributed. Thus, the neural network approach can solve optimization problems in running time at the orders of magnitude much faster than conventional optimization algorithms executed on general-purpose digital computers.

Since Hopfield and Tank published the milestone articles [19], [33], neural networks for solving optimization problems have been studied by many researchers. Numerous good results have been obtained (see [3], [4], [6]–[15], [20]–[23], [25]–[28], [30], [36]–[47], and the references therein), in particular, many of these models can be used to solve (1) or its special cases. Among them, Kennedy and Chua [23] proposed a primal neural network for solving nonlinear programming problems by combining the gradient method and the penalty function method. Because their network contains a penalty parameter and its energy function can be viewed as an "inexact" penalty function, it only generates approximate solutions and has an implementation problem when the penalty parameter is large. To avoid using penalty parameters, several elegant neural networks have been proposed. Rodríguez-Vázquez *et al.* [30] proposed a class of optimization neural networks. However, the right-hand side of their system is discontinuous, and its convergence requires the boundedness of the feasible region Ω of the problem and the condition that the interior set Ω^o of Ω is nonempty [15]. The models in [3], [6], and [39] can solve (1) without equality and inequality constraints. They all have one-layer structure and low complexity, yet the model in [3] can only solve the unconstrained case (except bounds), and it is unwise for using the models in [6] and [39] to solve (1) when either inequality or equality constraints are present since another quadratic convex programming problem with the same constraints must be solved to compute the projection on its feasible region. By extending the models in [6] and [39] to constrained optimization problems, Xia *et al.* developed several models with a one-layer structure (see [38], [41], [44]–[46], and the references therein). Even though these models can solve some nonconvex optimization problems, and be used to solve (1) by reformulating two-sided inequality constraints into one-sided ones, they might not be stable even when $f(x)$ is convex on R^n (see Example 2 in Section IV) and the model complexity must be increased substantially since the size of the resulting network must be enlarged to $2m + n + p$. When applied to (1) by the same transformation to two-sided inequality constraints, the models in [9], [12], [13], and [22] are stable and convergent when $f(x)$ is convex on R^n, yet they all have two-layer structure and may not be stable when $f(x)$ is convex

Manuscript received November 13, 2008; revised November 27, 2009 and March 02, 2010; accepted March 03, 2010. Date of publication April 12, 2010; date of current version June 03, 2010. This work was supported in part by the National Natural Science Foundation of China under Grants 60671063 and 10902062, and grants from Hong Kong Baptist University (FRG) and the Research Grant Council of Hong Kong.

X. B. Gao is with the College of Mathematics and Information Science, Shaanxi Normal University, Xi'an, Shaanxi 710062, China (e-mail: xinbaog@snnu.edu.cn).

L.-Z. Liao is with the Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong, China (e-mail: liliao@hkbu.edu.hk).

Digital Object Identifier 10.1109/TNN.2010.2045129

on $\mathcal{D} = \{x \in R^n | Dx = d\}$ (see Example 1 in Section IV). In particular, it should be pointed out that the model in [22] is an application of the model in [12] for (1) with one-sided inequality constraints since the former one can be derived from the latter one by using suitable variable substitution. The model in [8] and [10] can be directly applied to (1), yet they share the same drawbacks as the models in [9], [12], [13], and [22] except that its size is $m + n + p$. By using variables elimination, Hu and Wang [20] proposed a neural network for linear variational inequalities and problem (1). Their model only requires $m + n$ state variables and is globally convergent when $f(x)$ is convex on $\mathcal{D}$, yet it has still a two-layer structure and must transform some bounded constraints into new two-sided inequality ones. When inequality constraints are absent in (1), Xia [37] proposed a primal–dual neural network. This network has a two-layer structure and was further simplified by the models in [11], [14], and [34]. Obviously these models can solve (1) by adding slack variables to convert inequality constraints into equality constraints, yet they all have two-layer structure and the size of the resulting networks must be enlarged to $4m + n + p$. To reduce the network structure, several dual neural networks in [21], [26], [40], [43], and [47] are proposed to solve (1) with $A = I_n$ in [21] and A being positive definite (i.e., $f(x)$ is strictly convex on R^n) in the others by using variables substitution or variables elimination. Even though these models all have one-layer structure, they cannot be used for (1) with A being positive semidefinite only (i.e., $f(x)$ is only convex on R^n). In particular, Liu and Wang [28] developed a one-layer neural network for (1) without inequality constraints. Their model has a discontinuous hard-limiting activation function, and is stable and globally convergent by choosing some suitable parameters when $f(x)$ is strictly convex on $\mathcal{D}$. Even though the output equation of this model might not be defined when $f(x)$ is only convex on R^n, and the size of the resulting model for (1) is $2m + n$ by adding slack variables, yet their model motivates us to develop some new neural network for (1) which has a one-layer structure, least state variables, continuous right-hand side, and requires weak stability conditions.

Based on the above considerations, in this paper, we propose a new neural network for solving problem (1) by introducing some new vectors. The proposed neural network 1) has a one-layer structure, 2) is proved to be stable in the sense of Lyapunov, and converges to an exact optimal solution of problem (1) when $f(x)$ is convex on the set defined by equality constraints. Compared with existing one-layer neural networks for quadratic programming problems, the proposed neural network has least neurons and requires weak stability conditions. In particular, unlike the existing models in [27] and [28], the new model does not contain any adjustable parameter when applied to linear and convex quadratic programming, and thus avoids the difficulty of choosing network parameter and decreases the network complexity in implementation.

Throughout the paper, we assume that there exists a finite $x^* \in \mathcal{X}^* = \{x \in R^n \mid x \text{ is an optimal solution of (1)}\}$, and problem (1) is strong consistent (see [1, p. 97]), i.e., there exists an $x' \in R^n$ such that

$$\begin{cases} Dx' = d, \\ \underline{h}_j < (Bx')_j < \overline{h}_j, & \text{for } j = 1, 2, \ldots, m \\ \underline{l}_i < x'_i < \overline{l}_i, & \text{for } i = 1, 2, \ldots, n. \end{cases}$$

In our following discussions, we let $\|\cdot\|$ denote the Euclidean norm for both vectors and matrices, I_n denote the identity matrix of order n, $\underline{l} = (\underline{l}_1, \underline{l}_2, \ldots, \underline{l}_n)^T$, $\overline{l} = (\overline{l}_1, \overline{l}_2, \ldots, \overline{l}_n)^T$, $\underline{h} = (\underline{h}_1, \underline{h}_2, \ldots, \underline{h}_m)^T$, $\overline{h} = (\overline{h}_1, \overline{h}_2, \ldots, \overline{h}_m)^T$, $\mathcal{D} = \{x \in R^n | Dx = d\}$, $\mathcal{S} = \{x \in R^n | Dx = 0\}$, $\nabla\varphi(x) = (\partial\varphi(x)/\partial x_1, \partial\varphi(x)/\partial x_2, \ldots, \partial\varphi(x)/\partial x_n)^T \in R^n$ denote the gradient vector of the differentiable function $\varphi(x)$ at x. For any $n \times n$ real symmetric matrix M, $\lambda_{\min}(M)$ denotes its minimum eigenvalue. For any closed convex set $U \subset R^n$, $P_U : R^n \to U$ denotes the projection operator defined by

$$P_U(u) = \arg\min_{v \in U} \|u - v\|.$$

A basic property of the projection mapping [24] is

$$[v - P_U(v)]^T [P_U(v) - w] \geq 0 \qquad \forall v \in R^n, w \in U. \quad (2)$$

In particular, $P_{\mathcal{X}}(x) = [P_{\mathcal{X}}(x_1), P_{\mathcal{X}}(x_2), \ldots, P_{\mathcal{X}}(x_n)]^T$ with $P_{\mathcal{X}}(x_i) = \min\{\overline{l}_i, \max\{x_i, \underline{l}_i\}\}$ for $i = 1, 2, \ldots, n$, and $P_{\mathcal{B}}(\lambda) = [P_{\mathcal{B}}(\lambda_1), P_{\mathcal{B}}(\lambda_2), \ldots, P_{\mathcal{B}}(\lambda_m)]^T$ with $P_{\mathcal{B}}(\lambda_j) = \min\{\overline{h}_j, \max\{\lambda_j, \underline{h}_j\}\}$ for $j = 1, 2, \ldots, m$.

For the convenience of later discussions, it is necessary to introduce the following definition and lemma.

Definition 1: A neural network is said to be stable in the sense of Lyapunov, globally asymptotically stable, if the corresponding dynamical system is so.

Lemma 1 [20], [28]: $f(x)$ is convex (strictly convex) on $\mathcal{D}$ if and only if $x^T Ax \geq 0$ (> 0) for any $x \in \mathcal{S}$.

The rest of the paper is organized as follows. In Section II, a neural network for solving problem (1) is constructed. The stability and convergence of the proposed neural network are analyzed in Section III. Numerical simulations are provided in Section IV. Finally, some concluding remarks are drawn in Section V.

II. A Neural Network Model

In this section, we will construct a neural network to solve problem (1), and show the advantages of the new model over the existing ones. First, we state the following result for (1).

Theorem 1: If $x^* \in R^n$ is a solution of (1), then there exists a $(\lambda^*, \mu^*) \in R^{m+n}$ such that $(x^*, Bx^*) \in \mathcal{X} \times \mathcal{B}$ and

$$\begin{cases} (x - x^*)^T [\alpha(Ax^* + a) - B^T\lambda^* - D^T\mu^*] \geq 0, \ \forall x \in \mathcal{X} \\ Dx^* = d, \ (\lambda - Bx^*)^T\lambda^* \geq 0, \ \forall \lambda \in \mathcal{B} \end{cases} \quad (3)$$

where $\alpha > 0$ is a constant. Furthermore, if $f(x)$ is convex on $\mathcal{D}$, then $x^* \in \mathcal{X}^*$ if and only if there exists a $(\lambda^*, \mu^*) \in R^{m+n}$ such that (3) holds.

Proof: According to [2], we know that if x^* is a solution of (1), then

$$(x - x^*)^T (Ax^* + a) \geq 0 \qquad \forall x \in \Omega \quad (4)$$

where $\Omega = \{x \in \mathcal{X}, Bx \in \mathcal{B}, Dx = d\}$. Thus, $x^* \in \mathcal{X}$ is an optimal solution for the following linear programming problem:

$$\min\{\alpha x^T (Ax^* + a) | x \in \Omega\}.$$

Obviously the above problem can be rewritten as

$$\begin{cases} \min & \alpha x^T(Ax^*+a) \\ \text{s.t.} & Bx=\xi,\ Dx=d, \\ & x\in\mathcal{X},\ \xi\in\mathcal{B} \end{cases} \tag{5}$$

and its Lagrange function is

$$L(x,\xi,\lambda,\mu)=\alpha x^T(Ax^*+a)-\lambda^T(Bx-\xi)-\mu^T(Dx-d)$$

which is defined on $\Gamma = \mathcal{X}\times\mathcal{B}\times R^m\times R^p$. From the Kuhn–Tucker theorem in [1], we know that $(x^*,\xi^*)\in\mathcal{X}\times\mathcal{B}$ is a solution of problem (5) if and only if there exists a $(\lambda^*,\mu^*)\in R^m\times R^p$ such that $(x^*,\xi^*,\lambda^*,\mu^*)$ is a saddle point of $L(x,\xi,\lambda,\mu)$ on Γ, that is

$$L(x^*,\xi^*,\lambda,\mu)\le L(x^*,\xi^*,\lambda^*,\mu^*)\le L(x,\xi,\lambda^*,\mu^*) \quad \forall(x,\xi,\lambda,\mu)\in\Gamma.$$

Using these inequalities, (3) can be obtained easily.

If $f(x)$ is convex on $\mathcal{D}$, then problem (1) is convex on Ω. Thus, $x^*\in\mathcal{X}^*$ if and only if it is a solution of (4) if and only if there exists a $(\lambda^*,\mu^*)\in R^m\times R^p$ such that (3) holds. ■

From (2) and (3), we can easily verify the following result.

Lemma 2: If $x^*\in R^n$ is a solution of (1), then there exists a $(\lambda^*,\mu^*)\in R^{m+p}$ such that

$$\begin{cases} x^*=P_{\mathcal{X}}[x^*-\alpha(Ax^*+a)+B^T\lambda^*+D^T\mu^*] \\ Bx^*=P_{\mathcal{B}}(Bx^*-\lambda^*),\ \ Dx^*=d. \end{cases} \tag{6}$$

Furthermore, if $f(x)$ is convex on $\mathcal{D}$, then $x^*\in\mathcal{X}^*$ if and only if there exists a $(\lambda^*,\mu^*)\in R^{m+p}$ such that (6) holds.

Lemma 2 indicates that (x^*,Bx^*) is the projection of some vectors on $\mathcal{X}\times\mathcal{B}$ if x^* is a solution of (1).

Lemma 3: Let $G=I_n+\alpha A+B^TB$, then the following two results hold.

i) G is positive definite if and only if $1+\lambda_{\min}(\alpha A+B^TB)>0$.

ii) G is positive definite if $\lambda_{\min}(A+B^TB)<0$ and $0<\alpha<\min\{1,-1/\lambda_{\min}(A+B^TB)\}$.

Proof:

i) Since $\alpha A+B^TB$ is symmetric, there exist an orthogonal matrix $V\in R^{n\times n}$ and a diagonal matrix $\Lambda=\mathrm{diag}(\lambda_1(\alpha A+B^TB),\lambda_2(\alpha A+B^TB),\ldots,\lambda_n(\alpha A+B^TB))$ such that $\alpha A+B^TB=V\Lambda V^T$, where $\lambda_i(\alpha A+B^TB)$ is the i-th eigenvalue of $\alpha A+B^TB$ for $i=1,2,\ldots,n$. Then, $G=V(I_n+\Lambda)V^T$. Thus, G is positive definite if and only if $1+\lambda_{\min}(\alpha A+B^TB)>0$.

ii) Suppose that G is not positive definite, then there exists an $x\in R^n$ with $x\neq 0$ such that $x^TGx\le 0$, i.e.,

$$\|x\|^2+x^T(\alpha A+B^TB)x\le 0.$$

Thus

$$(1-\alpha)\|Bx\|^2+[1+\alpha\lambda_{\min}(A+B^TB)]\|x\|^2\le 0.$$

From the assumption, $1-\alpha>0$ and $1+\alpha\lambda_{\min}(A+B^TB)>0$. These and the above inequality imply that $x=0$, which is impossible since $x\neq 0$. This completes the proof. ■

In order to construct a new neural network model for (1), we introduce the following variables:

$$\begin{cases} y^*=x^*+\alpha(Ax^*+a)-B^T\lambda^*-D^T\mu^* \\ z^*=Bx^*+\lambda^*. \end{cases} \tag{7}$$

By substituting $\lambda^*=z^*-Bx^*$ into the first equality in (7), we obtain

$$Gx^*=y^*+B^Tz^*-\alpha a+D^T\mu^* \tag{8}$$

where $G=I_n+\alpha A+B^TB$. Now, we fix α such that we have (9) shown at the bottom of the page. Then, G is symmetric and positive definite from Lemma 3 since $\lambda_{\min}(A+B^TB)\le -1$ from Lemma 3i) when I_n+A+B^TB is not positive definite. It follows from (8) that

$$x^*=G^{-1}(y^*+B^Tz^*-\alpha a+D^T\mu^*). \tag{10}$$

This and $Dx^*=d$ imply that

$$DG^{-1}(y^*+B^Tz^*-\alpha a+D^T\mu^*)=d.$$

Since $\mathrm{rank}(D)=p\le n$, $DG^{-1}D^T$ is also symmetric and positive definite. Then

$$\mu^*=(DG^{-1}D^T)^{-1}[d-DG^{-1}(y^*+B^Tz^*-\alpha a)].$$

It follows by substituting this equality into (10) that

$$x^*=C(y^*+B^Tz^*)+q$$

where

$$\begin{cases} G=I_n+\alpha A+B^TB \\ C=G^{-1}-G^{-1}D^T(DG^{-1}D^T)^{-1}DG^{-1} \\ q=G^{-1}D^T(DG^{-1}D^T)^{-1}d-\alpha Ca. \end{cases} \tag{11}$$

By substituting x^* and $\lambda^*=z^*-Bx^*$ into (6), we have

$$\begin{cases} C(y^*+B^Tz^*)+q=P_{\mathcal{X}}[2(C(y^*+B^Tz^*)+q)-y^*] \\ B[C(y^*+B^Tz^*)+q]=P_{\mathcal{B}}[2B(C(y^*+B^Tz^*)+q)-z^*]. \end{cases}$$

For simplicity, we denote $R=BC$, $S=RB^T$, and $s=Bq$, where C, G, and q are defined in (11) and α is defined in (9). Then, the above results suggest the following neural network for problem (1):

$$\alpha\begin{cases} =1, & \text{if } I_n+A+B^TB \text{ is positive definite,} \\ \in\left(0,\dfrac{-1}{\lambda_{\min}(A+B^TB)}\right), & \text{otherwise} \end{cases} \tag{9}$$

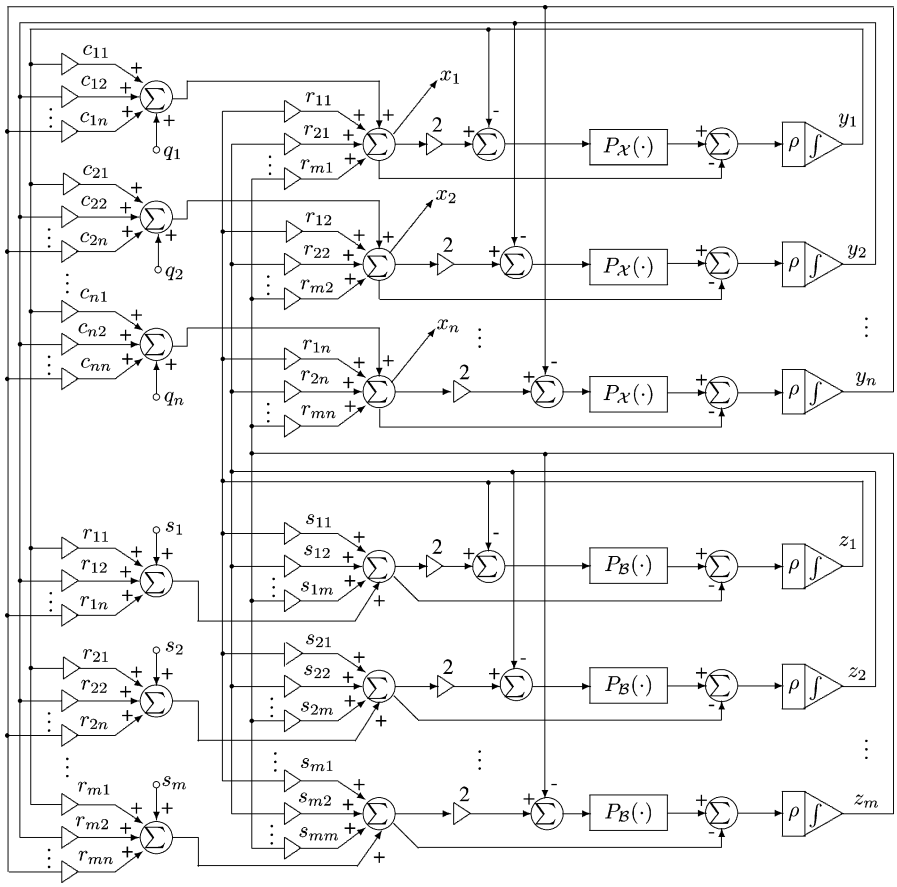


Fig. 1. Architecture of network (12)–(13).

- state equation

$$\frac{d}{dt}\begin{pmatrix} y \\ z \end{pmatrix} = -\rho \begin{pmatrix} Cy + R^T z + q - P_{\mathcal{X}}[2(Cy + R^T z + q) - y] \\ Ry + Sz + s - P_{\mathcal{B}}[2(Ry + Sz + s) - z] \end{pmatrix}; \tag{12}$$

- output equation

$$x = Cy + R^T z + q; \tag{13}$$

where $\rho > 0$ is a scaling constant.

It is easy to see that $DC = 0$ and $Dq = d$, where C and q are defined in (11). Then, the output vector x defined in (13) always satisfies equality constraints $Dx = d$. The architecture of neural network (12)–(13) is shown in Fig. 1, where vector x is neural network's output, vectors $q = (q_1, q_2, \ldots, q_n)^T$ and $s = (s_1, s_2, \ldots, s_m)^T$ are the external inputs, and the other parameters are defined by $C = (c_{ij})_{n\times n}$, $R = (r_{ij})_{m\times n}$, and $S = (s_{ij})_{m\times m}$. The projection operators $P_{\mathcal{X}}(\cdot)$ and $P_{\mathcal{B}}(\cdot)$ could be easily implemented by using piecewise-activation functions [3]. According to Fig. 1, the circuit realizing the proposed neural network (12)–(13) consists of $m+n$ integrators, $(m+n)^2$ connection weights, $m+n$ activation functions for $P_{\mathcal{X}}(\cdot)$ and $P_{\mathcal{B}}(\cdot)$, some amplifiers, and adders. Thus, it can be implemented by using simple hardware units.

For the choice of α, we have the following remark.

Remark 1:

i) Since $|\lambda_{\min}(A + B^T B)| \leq \|A + B^T B\|_1$, we can let $\alpha \in (0, 1/\|A + B^T B\|_1)$ in (9) when $I_n + A + B^T B$ is not positive definite, where $\|A\|_1 = \max_j \sum_{i=1}^n |a_{ij}|$.

ii) Obviously G defined in (11) is positive definite when $I_n + \alpha A$ is so. Then, we can simply choose $\alpha \in (0, 1/\|A\|_1)$ in (9) when $I_n + A$ is not positive definite.

To show the advantages of the proposed neural network (12)–(13), we compare it with six existing neural networks (three one-layer models and three two-layer ones). First, let us look at the models in [38] and [46], whose state equations for (1) can be written in details as

$$\frac{d}{dt}\begin{pmatrix} x \\ \lambda \\ \mu \end{pmatrix} = -\rho \begin{pmatrix} x - P_\Omega(x - Ax - a + \hat{B}^T\lambda + D^T\mu) \\ \lambda - (\lambda - \hat{B}x + b)^+ \\ Dx - d \end{pmatrix} \tag{14}$$

and

$$\frac{d}{dt}\begin{pmatrix} y \\ \lambda \\ \mu \end{pmatrix} = -\rho \begin{pmatrix} y - P_\Omega(y) + AP_\Omega(y) + a - \hat{B}^T\lambda^+ - D^T\mu \\ \lambda - \lambda^+ + \hat{B}P_\Omega(y) - b \\ DP_\Omega(y) - d \end{pmatrix} \tag{15}$$

respectively, where $x, y \in R^n$, $\lambda \in R^{2m}$, $\mu \in R^p$, $\lambda^+ = (\lambda_1^+, \lambda_2^+, \ldots, \lambda_{2m}^+)^T$ with $\lambda_i^+ = \max\{0, \lambda_i\}$ for $i = 1, 2, \ldots, 2m$, $B = (B^T, -B^T)^T$, and $b = (\underline{h}^T, \overline{h}^T)^T$. The corresponding output equations are x and $x = P_\Omega(y)$, respectively. Thus, they

all have $2m+n+p$ state variables, while the proposed model (12)–(13) has $m+n$ state variables. More importantly, the stability of both (14) and (15) requires the positive definiteness of A (i.e., the strict convexity of f on R^n), while model (12)–(13) will be shown to be asymptotically stable when $f(x)$ is convex on $\mathcal{D}$ (see Theorem 3 in Section III). Thus, both (14) and (15) could not be used to solve (1) when A is positive semidefinite (see Example 2 in Section IV), while Theorem 3 ensures the stability and convergence of the proposed network for this case.

Next, we compare the proposed model with the model in [28], which can be written as:

- state equation

$$\frac{dw}{dt} = -\rho\{(I_{n+2m}-\hat{P})w+[(I_{n+2m}-\hat{P})\hat{A}+\zeta\hat{P}]g(w)-\hat{q}\}; \quad (16)$$

- output equation

$$z = [(I_{n+2m}-\hat{P})\hat{A}+\zeta\hat{P}]^{-1}[\hat{q}-(I_{n+2m}-\hat{P})w]; \quad (17)$$

where $\zeta>0$ is a parameter, $w=(x^T,\xi^T,\eta^T)^T$, $\hat{d}=(\underline{h}^T,\overline{h}^T, d^T)^T$, $\hat{P}=\hat{D}^T(\hat{D}\hat{D}^T)^{-1}\hat{D}$, $\hat{q}=-\hat{a}+\hat{P}\hat{a}+\zeta\hat{D}^T(\hat{D}\hat{D}^T)^{-1}\hat{d}$, $\hat{a}=(a^T,0^T,0^T)^T$

$$\hat{A}=\begin{pmatrix} A & O & O \\ O & O & O \\ O & O & O \end{pmatrix} \qquad \hat{D}=\begin{pmatrix} B & -I_m & O \\ O & B & I_m \\ D & O & O \end{pmatrix}$$

with O denoting the zero matrix with proper dimensions

$$g_i(w_i)\begin{cases} =\overline{l}_i, & \text{if } w_i>0 \\ \in[\underline{l}_i,\overline{l}_i], & \text{if } w_i=0 \\ =\underline{l}_i, & \text{if } w_i<0 \end{cases}, \quad \text{for } i=1,2,\ldots,n \text{ and}$$

$$g_i(w_i)\begin{cases} \in[0,+\infty), & \text{if } w_i=0 \\ =0, & \text{if } w_i<0 \end{cases}, \quad \text{for } i=n+1,\ldots,n+2m.$$

Thus, both (12) and (16) have one-layer structure, yet the size, definition, and stability for them are different. First, (16) has $2m+n$ state variables, while (12) has $m+n$ state variables. Second, the right-hand side of (16) is discontinuous and there might be no parameter ζ such that its output (17) can be defined when $f(x)$ is only convex on $\mathcal{D}$ (see Example 1 in Section IV), while the right-hand side of (12) is continuous and the output (13) is well defined for this case. Third, network (16)–(17) is proven to be globally convergent to optimal solutions as long as $f(x)$ is strictly convex on $\mathcal{D}$, while the stability and convergence of the proposed neural network requires only the convexity of f on $\mathcal{D}$. Thus, (16)–(17) could not be used to (1) when f is only convex on $\mathcal{D}$ (see Example 1 in Section IV). Finally, the proposed neural network will be guaranteed to be asymptotically stable without any parameter when A is positive semidefinite, while (16) needs to choose the parameter ζ to ensure the stability and convergence even when A is positive definite. In particular, (16) cannot be applied for linear programming problems.

Third, we compare the proposed neural network model with the models in [9] and [13], whose state equations for (1) are shown in (18) and (19) at the bottom of the page, where $x,y\in R^n$, $\lambda\in R^{2m}$, $\mu\in R^p$, $\hat{\lambda}=(\lambda-\hat{B}x+b)^+$, $\tilde{\lambda}=(\lambda-\hat{B}P_\Omega(y)+b)^+$, $\hat{B}$, and b are defined in (14)–(15). Thus, they all have two-layer structure and $2m+n+p$ state variables, while (12)–(13) has a one-layer structure and $m+n$ state variables. Moreover, the stability of both (18) and (19) requires the positive semidefiniteness of A, while model (12)–(13) requires only the convexity of f on $\mathcal{D}$ (see Theorem 3 in Section III). Thus, (18)–(19) could not be used to solve (1) when $f(x)$ is only convex on $\mathcal{D}$ (see Example 1 in Section IV).

Finally, let us look at the model in [20]. Since $\mathrm{rank}(D)=p$, there exists a permutation matrix $P\in R^{n\times n}$ such that $DP^T=(D_I,D_{II})$ with $D_I\in R^{p\times p}$ being nonsingular and $D_{II}\in R^{p\times(n-p)}$. Then, the model in [20] for (1) can be written as:

- state equation

$$\frac{d}{dt}\begin{pmatrix} u \\ v \end{pmatrix}=\rho\begin{pmatrix} I_{n-p}+\bar{A} & \bar{B}^T \\ -\bar{B} & I_{m+p} \end{pmatrix} \times\begin{pmatrix} P_{\mathcal{U}}(u-\bar{A}u+\bar{B}^Tv-\bar{b})-u \\ P_{\mathcal{V}}(\bar{B}u-v)-\bar{B}u \end{pmatrix}; \quad (20)$$

- output equation

$$x=P^T(Qu+q); \quad (21)$$

where $\bar{A}=Q^TPAP^TQ$, $\bar{b}=Q^TP(AP^Tq+a)$, $\mathcal{U}=\{u\in R^{n-p}|(P\underline{l})_{p+i}\le u_i\le(P\overline{l})_{p+i}\text{ for } i=1,2,\ldots,n-p\}$, $\underline{x}_I=((P\underline{l})_1,(P\underline{l})_2,\ldots,(P\underline{l})_p)^T$, $\overline{x}_I=((P\overline{l})_1,(P\overline{l})_2,\ldots,(P\overline{l})_p)^T$

$$Q=\begin{pmatrix} -D_I^{-1}D_{II} \\ I_{n-p} \end{pmatrix} \quad q=\begin{pmatrix} D_I^{-1}d \\ 0 \end{pmatrix} \quad \bar{B}=\begin{pmatrix} BP^TQ \\ -D_I^{-1}D_{II} \end{pmatrix}$$

and

$$\mathcal{V}=\left\{v\in R^{m+p}|\begin{pmatrix} \underline{h}-BP^Tq \\ \underline{x}_I-D_I^{-1}d \end{pmatrix}\le v\le\begin{pmatrix} \overline{h}-BP^Tq \\ \overline{x}_I-D_I^{-1}d \end{pmatrix}\right\}.$$

Even though this model and (12) have the same size and stability requirement on f, unlike (12)–(13), (20)–(21) has a two-layer structure and requires to partition D and transform $n-p$ bounded constraints into two-sided inequality constraints.

To see more clearly, a comparison of the proposed neural network with the above six models and additional seven models

$$\frac{d}{dt}\begin{pmatrix} x \\ \lambda \\ \mu \end{pmatrix}=-\rho\begin{pmatrix} 2[x-P_\Omega(x-Ax-a+\hat{B}^T\hat{\lambda}+D^T(\mu-Dx+d))] \\ \lambda-\hat{\lambda} \\ Dx-d \end{pmatrix} \quad (18)$$

and

$$\frac{d}{dt}\begin{pmatrix} y \\ \lambda \\ \mu \end{pmatrix}=-\rho\begin{pmatrix} y-P_\Omega(y)+AP_\Omega(y)+a-\hat{B}^T\tilde{\lambda}-D^T[\mu-DP_\Omega(y)+d] \\ \lambda-\tilde{\lambda} \\ DP_\Omega(y)-d \end{pmatrix} \quad (19)$$

TABLE I
COMPARISON OF RELATED NEURAL NETWORKS IN TERMS OF THE MODEL COMPLEXITY AND THE STABILITY CONDITIONS FOR (1)

model	layer(s)	neurons	condition on f
model in [34]	2	$4m+n+p$	convex on R^n
model in [37]	2	$4m+n+p$	convex on R^n
(18)	2	$2m+n+p$	convex on R^n
(19)	2	$2m+n+p$	convex on R^n
model in [22]	2	$2m+n+p$	convex on R^n
model in [10]	2	$m+n+p$	convex on R^n
(20)-(21)	2	$m+n$	convex on $\mathcal{D}$
(14)	1	$2m+n+p$	strictly convex on R^n
(15)	1	$2m+n+p$	strictly convex on R^n
model in [40], [43]	1	$m+n+p$	strictly convex on R^n
model in [25]	1	$4m+n$	convex on R^n
(16)-(17)	1	$2m+n$	strictly convex on $\mathcal{D}$
model in [26]	1	$m+n$	strictly convex on R^n
(12)-(13)	1	$m+n$	convex on $\mathcal{D}$

is provided in Table I. From Table I, we see that the proposed neural network (12)–(13) has least neurons and requires weak stability condition.

The following result reveals the relationship between an equilibrium point of (12) and a solution of (1).

Lemma 4:

i) There exists a $(y^*, z^*) \in R^{m+n}$ such that (y^*, z^*) is an equilibrium point of (12).

ii) If $f(x)$ is convex on $\mathcal{D}$ and (y^*, z^*) is an equilibrium point of (12), then $Cy^* + R^T z^* + q \in \mathcal{X}^*$, where $R = BC$, C, and q are defined in (11).

Proof:

i) From the assumption, $\mathcal{X}^* \neq \emptyset$. Let $x^* \in \mathcal{X}^*$, then there exists a $(\lambda^*, \mu^*) \in R^{m+p}$ such that (6) holds from Theorem 1. From the above analysis, we know that (y^*, z^*) defined in (7) is an equilibrium point of (12).

ii) Let $x^* = Cy^* + R^T z^* + q$, then

$$\begin{cases} x^* = P_{\mathcal{X}}(2x^* - y^*) \\ Bx^* = P_{\mathcal{B}}(2Bx^* - z^*) \end{cases} \tag{22}$$

from (12), and $Dx^* = d$ since $DC = 0$ and $Dq = d$. Thus, $(x^*, Bx^*) \in \mathcal{X} \times \mathcal{B}$.

On the other hand, from (11), we have

$$\begin{cases} D^T(DG^{-1}D^T)^{-1}DG^{-1} = I_n - GC \\ D^T(DG^{-1}D^T)^{-1}d = G(q + \alpha Ca). \end{cases}$$

Let $\lambda^* = z^* - Bx^*$ and $\mu^* = (DG^{-1}D^T)^{-1}[d - DG^{-1}(y^* + B^T z^* - \alpha a)]$, then

$$\begin{aligned} D^T\mu^* &= G(q + \alpha Ca) + (GC - I_n)(y^* + B^T z^* - \alpha a) \\ &= G(q + Cy^* + R^T z^*) - y^* - B^T(\lambda^* + Bx^*) + \alpha a \\ &= (G - B^T B)x^* - y^* - B^T\lambda^* + \alpha a \\ &= (I_n + \alpha A)x^* - y^* - B^T\lambda^* + \alpha a. \end{aligned}$$

That is

$$y^* = x^* + \alpha(Ax^* + a) - B^T\lambda^* - D^T\mu^*.$$

This, (22), and $\lambda^* = z^* - Bx^*$ imply that (6) holds. Since $f(x)$ is convex on $\mathcal{D}$, $x^* \in \mathcal{X}^*$ from Lemmas 1 and 2. ■

III. STABILITY ANALYSIS

In this section, we will study some stability and convergence properties for (12)–(13). First, we prove the following lemma which is very useful in our later discussion.

Lemma 5: Let C be defined in (11), then $C - C^2 - CB^TBC = \alpha CAC$.

Proof: Since $G = I_n + \alpha A + B^T B$ defined in (11) is symmetric and positive definite, there exists a symmetric and positive-definite matrix $F \in R^{n\times n}$ such that $G = F^2$. Then, $G^{-1} = F^{-2}$. It follows from (11) that

$$\begin{aligned} C &= F^{-2} - F^{-2}D^T(DF^{-2}D^T)^{-1}DF^{-2} \\ &= F^{-1}[I_n - F^{-1}D^T(DF^{-2}D^T)^{-1}DF^{-1}]F^{-1} \\ &= F^{-1}PF^{-1} \end{aligned}$$

where $P = I_n - F^{-1}D^T(DF^{-2}D^T)^{-1}DF^{-1}$. It is easy to see that $P^T = P$ and $P^2 = P$. Thus

$$\begin{aligned} C &= F^{-1}P^2F^{-1} = F^{-1}PF^{-1}F^2F^{-1}PF^{-1} \\ &= CF^2C = CGC. \end{aligned}$$

Therefore

$$C - C^2 - CB^TBC = C(G - I_n - B^TB)C = \alpha CAC.$$

This completes the proof. ■

The result in Lemma 5 is very important and paves a way for the stability result of neural network (12)–(13). In particular, neural network (12)–(13) has the following basic property.

Theorem 2: $\forall u^0 = ((y^0)^T, (z^0)^T)^T \in R^{m+n}$, there exists a unique continuous solution $u(t) = ((y(t))^T, (z(t))^T)^T \in R^{m+n}$ of (12) with $u(0) = u^0$ for all $t \geq 0$.

Proof: For simplicity, we let $E(u)$ denote the right-side hand of (12). From (2), we have

$$\begin{cases} \|P_{\mathcal{X}}(x) - P_{\mathcal{X}}(x')\| \leq \|x - x'\|, \ \forall x, x' \in R^n \\ \|P_{\mathcal{B}}(z) - P_{\mathcal{B}}(z')\| \leq \|z - z'\|, \ \forall z, z' \in R^m. \end{cases}$$

Then, for any $u, u' = ((x')^T, (y')^T)^T \in R^{m+n}$, from the above inequalities, we have

$$\begin{aligned} &\|E(u) - E(u')\| \\ &\quad \leq \|C(y - y') + R^T(z - z')\| \\ &\qquad + \|R(y - y') + S(z - z')\| \\ &\qquad + \|P_{\mathcal{X}}[2(Cy + R^Tz + q) - y] \\ &\qquad - P_{\mathcal{X}}[2(Cy' + R^Tz' + q) - y']\| \\ &\qquad + \|P_{\mathcal{B}}[2(Ry + Sz + s) - z] \\ &\qquad - P_{\mathcal{B}}[2(Ry' + Sz' + s) - z']\| \\ &\quad \leq (\|C\| + \|R\|)\|y - y'\| + (\|R\| + \|S\|)\|z - z'\| \\ &\qquad + \|(2C - I)(y - y') + 2R^T(z - z')\| \\ &\qquad + \|2R(y - y') + (2S - I)(z - z')\| \\ &\quad \leq (1 + 3\|C\| + 3\|R\|)\|y - y'\| \\ &\qquad + (1 + 3\|R\| + 3\|S\|)\|z - z'\|. \end{aligned}$$

This implies that $E(u)$ is Lipschitz continuous on R^{m+n}. Thus, the result can be established from [16, Th. 1]. ■

The results of Lemma 4 and Theorem 2 indicate that neural network (12)–(13) is well defined. Now we are ready to state and prove the following stability result for (12)–(13).

Theorem 3: If $f(x)$ is convex on $\mathcal{D}$, then neural network (12)–(13) is stable in the sense of Lyapunov, and for any $u^0 = ((y^0)^T, (z^0)^T)^T$, its state trajectory $u(t) = ((y(t))^T, (z(t))^T)^T$ and output trajectory $x(t)$ will converge to an equilibrium point of (12) and an optimal solution of (1), respectively. In particular, (12)–(13) is globally asymptotically stable when (12) has a unique equilibrium point.

Proof: From Theorem 2, $\forall u^0 \in R^{n+m}$, system (12) has a unique continuous solution $u(t) \in R^{n+m}$ with $u(0) = u^0$ for all $t \geq 0$.

For simplicity, we denote $E_1(u) = Cy + R^T z + q - P_{\mathcal{X}}[2(Cy + R^T z + q) - y]$ and $E_2(u) = Ry + Sz + q - P_{\mathcal{B}}[2(Ry + Sz + q) - z]$. Let $u^* = (y^*, z^*)$ be an equilibrium point of (12), then $E_1(u^*) = 0$ and $E_2(u^*) = 0$, and $Cy^* + R^T z^* + q \in \mathcal{X}^*$ from Lemma 4ii). Thus

$$(x - Cy^* - R^T z^* - q)^T (y^* - Cy^* - R^T z^* - q) \geq 0 \quad \forall x \in \mathcal{X}.$$

This and $P_{\mathcal{X}}[2(Cy + R^T z + q) - y] \in \mathcal{X}$ imply that

$$[C(y - y^*) + R^T(z - z^*) - E_1(u)]^T (y^* - Cy^* - R^T z^* - q) \geq 0.$$

By substituting $v = 2(Cy + R^T z + q) - y$ and $w = Cy^* + R^T z^* + q$ into (2), we have

$$[E_1(u) + Cy + R^T z + q - y]^T \times [C(y - y^*) + R^T(z - z^*) - E_1(u)] \geq 0.$$

It follows from adding the above two inequalities that

$$[E_1(u) + (C - I)(y - y^*) + R^T(z - z^*)]^T \times [C(y - y^*) + R^T(z - z^*) - E_1(u)] \geq 0.$$

This and $R = BC$ imply that

$$\begin{aligned}(y - y^*)^T E_1(u) &\geq (y - y^*)^T [C(y - y^*) + R^T(z - z^*)] \\ &\quad + \|E_1(u)\|^2 - \|C(y - y^*) + R^T(z - z^*)\|^2 \\ &= \|E_1(u)\|^2 + (y - y^*)^T C\eta - \|C\eta\|^2 \end{aligned} \quad (23)$$

where $\eta = y - y^* + B^T(z - z^*)$. Noting that $S = BCB^T$, by the similar argument, we have

$$\begin{aligned}(z - z^*)^T E_2(u) &\geq (z - z^*)^T [R(y - y^*) + S(z - z^*)] \\ &\quad + \|E_2(u)\|^2 - \|R(y - y^*) + S(z - z^*)\|^2 \\ &= \|E_2(u)\|^2 + (z - z^*)^T BC\eta - \|BC\eta\|^2. \end{aligned} \quad (24)$$

Now, we consider the function

$$V(u, u^*) = \frac{1}{2}\|u - u^*\|^2.$$

From (12)–(13), (23)–(24), and Lemma 5, we have

$$\begin{aligned}&\frac{d}{dt}V[u(t), u^*] \\ &= -\rho[(y - y^*)^T E_1(u) + (z - z^*)^T E_2(u)] \\ &\leq -\rho[\|E_1(u)\|^2 + \|E_2(u)\|^2 + \eta^T(C - C^2 - CB^T BC)\eta] \\ &\leq -\rho(\|E_1(u)\|^2 + \|E_2(u)\|^2 + \alpha\eta^T CAC\eta) \quad \forall t \geq 0. \end{aligned} \quad (25)$$

On the other hand, $x^T Ax \geq 0$ for all $x \in \mathcal{S}$ from the convexity of $f(x)$ on $\mathcal{D}$ and Lemma 1. Since $DC = 0$, $Cx \in \mathcal{S}$ for all $x \in R^n$. Thus, $x^T CACx \geq 0$ for all $x \in R^n$. In particular, $\eta^T CAC\eta \geq 0$. This and (25) imply that

$$\frac{d}{dt}V[u(t), u^*] \leq -\rho(\|E_1(u)\|^2 + \|E_2(u)\|^2) \leq 0 \quad \forall t \geq 0.$$

Thus, system (12)–(13) is Lyapunov stable from [31, Th. 3.2]. The rest of the proof is similar to [9, Th. 3]. ■

Remark 2: When A is positive definite, let

$$y^* = Ax^* + a - B^T\lambda^* - D^T\mu^*$$

then (6) becomes

$$\begin{cases} C(y^* + B^T\lambda^*) + q = P_{\mathcal{X}}[C(y^* + B^T\lambda^*) + q - y^*] \\ B[C(y^* + B^T\lambda^*) + q] = P_{\mathcal{B}}[B(C(y^* + B^T\lambda^*) + q) - \lambda^*] \end{cases}$$

where

$$\begin{cases} C = A^{-1} - A^{-1}D^T(DA^{-1}D^T)^{-1}DA^{-1} \\ q = A^{-1}D^T(DA^{-1}D^T)^{-1}d - Ca. \end{cases} \quad (26)$$

That is $G = A$ in (11). Thus, a neural network for problem (1) becomes:

- state equation

$$\frac{d}{dt}\begin{pmatrix} y \\ z \end{pmatrix} = -\rho\begin{pmatrix} Cy + R^T\lambda + q - P_{\mathcal{X}}(Cy + R^T\lambda + q - y) \\ Ry + S\lambda + s - P_{\mathcal{B}}(Ry + S\lambda + s - \lambda) \end{pmatrix}; \quad (27)$$

- output equation

$$x = Cy + R^T\lambda + q; \quad (28)$$

where $\rho > 0$ is a scaling constant, $R = BC$, $S = RB^T$, $s = Bq$, C, and q are defined in (26). Furthermore, if $\mathcal{X} = R^n$, then (27)–(28) can be simplified as:

- state equation

$$\frac{dz}{dt} = -\rho[S\lambda + s - P_{\mathcal{B}}(S\lambda + s - \lambda)];$$

- output equation

$$x = R^T\lambda + q;$$

where ρ, S, q, and R are the same as in (27)–(28). This is just the model in [26]. Therefore, the idea to build model (12)–(13) can be used to derive some existing models.

Remark 3: The above results hold for any closed convex set. In particular, the common case for $\mathcal{X}$ or $\mathcal{B}$ is $\{x \in R^n \mid \|x\| \leq c\}$ (ball or l_2 norm constraint).

Remark 4: If A is positive semidefinite, then the dual problem of problem (1)

$$\begin{cases} \max & [f(x) - \mu^T(Dx-d) - \underline{\lambda}^T(Bx-\underline{h}) \\ & -\overline{\lambda}^T(\overline{h}-Bx) - \underline{\nu}^T(x-\underline{l}) - \overline{\nu}^T(\overline{l}-x)] \\ \text{s.t.} & Ax + a - D^T\mu + B^T(\overline{\lambda}-\underline{\lambda}) = \underline{\nu} - \overline{\nu}, \\ & \underline{\lambda} \geq 0, \quad \overline{\lambda} \geq 0, \quad \underline{\nu} \geq 0, \quad \overline{\nu} \geq 0 \end{cases}$$

can be solved by neural network (12)–(13). In fact, if (y^*, z^*) is an equilibrium point of (12), then it is easy to prove that $(x^*, \mu^*, (z^* - Bx^*)^+, (Bx^* - z^*)^+, (y^* - x^*)^+, (x^* - y^*)^+)$ is an optimal solution of the above dual problem, where $x^* = Cy^* + R^Tz^* + q$ and $\mu^* = (DG^{-1}D^T)^{-1}[d - DG^{-1}(y^* + B^Tz^* - a)]$.

IV. Illustrative Examples

In this section, four examples are provided to illustrate both the theoretical results achieved in Section III and the simulation results of the proposed model (12)–(13). The simulation is conducted in Matlab, and the ode solver used is ODE23S which is a stiff medium order method. For simplicity, we denote $e_n = (1, 1, \ldots, 1)^T \in R^n$ in the following.

Example 1 shows that the models in [9], [10], [12], [13], and [46] are unstable and the model in [28] is not well defined when $f(x)$ is only convex on $\mathcal{D}$.

Example 1: Consider the following quadratic programming problem, which is constructed according to [28, Example 1]:

$$\begin{cases} \min & f(x) = -5.25x_1^2 + x_2^2 + 2x_1x_2 + 6x_1 - 2x_2 \\ \text{s.t.} & 3x_1 - 2x_2 = 1, \quad 0 \leq x_1, x_2 \leq 10. \end{cases}$$

Then, this problem has a unique optimal solution $x^* = (1/3, 0)^T$, $\mathcal{X} = \{x \in R^2 | 0 \leq x_1, x_2 \leq 10\}$, $D = (3, -2)$, $d = 1$

$$A = \begin{pmatrix} -10.5 & 2 \\ 2 & 2 \end{pmatrix} \quad a = \begin{pmatrix} 6 \\ -2 \end{pmatrix}.$$

Although A is indefinite, $f(x)$ is convex on $\mathcal{D} = \{x \in R^2 | 3x_1 - 2x_2 = 1\}$ since $f(2x_2/3 + 1/3, x_2) = x_2/3 + 17/12$. Then, the proposed model (12)–(13) can be applied to this problem from the analysis in Section III. In this case, (12)–(13) can be simplified as:

- state equation

$$\frac{dy}{dt} = \rho\{P_{\mathcal{X}}[2(Cy+q) - y] - Cy - q\}; \tag{29}$$

- output equation

$$x = Cy + q; \tag{30}$$

where $\rho > 0$ is a scaling constant, and C and q are defined in (11) with $G = I_n + \alpha A$. Since $\lambda_{\min}(A) = -10.8122$, we let $\alpha = 1/11$, then $I_2 + \alpha A$ is positive definite from Lemma 3ii). All simulation results show that the state variables and output variables of (29)–(30) are asymptotically stable at $y^* = (1/3, 1/33)$ and x^*, respectively. For example, Figs. 2 and 3 depict the state trajectories and output trajectories of (29)–(30) with 50 random initial points and $\rho = 10$, respectively.

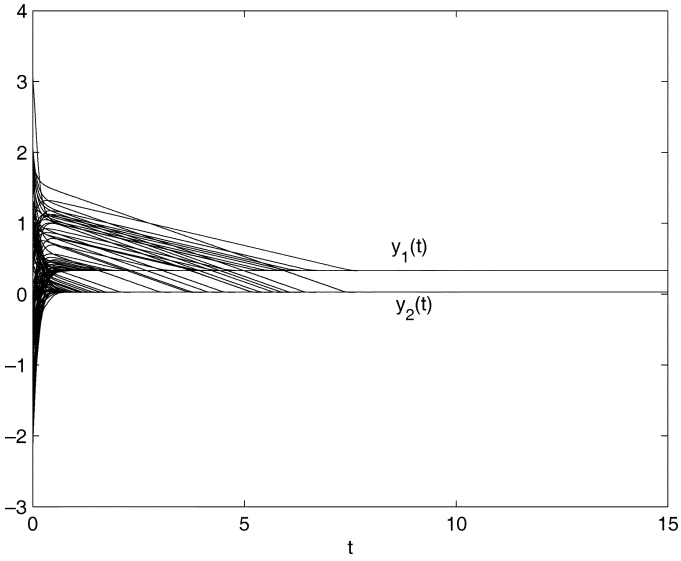


Fig. 2. Transient behavior of the state trajectories of (29) with 50 random initial points for Example 1.

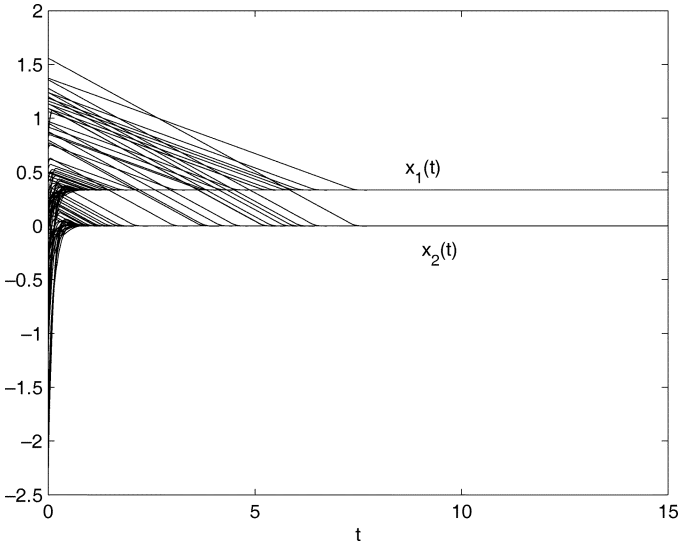


Fig. 3. Transient behavior of the output trajectories of (30) with 50 random initial points for Example 1.

To make a comparison, we first solve this problem using the models (18) and (19). In this case, $m = 0$ and the number of state variables for them is three. But Figs. 4 and 5 clearly show that (18) and (19) with initial point $(1, 5, 10)^T$ and $\rho = 10$ are unstable, respectively.

It should be mentioned that the models in [10]–[12] for Example 1 are also unstable since they are same as (18) with $m = 0$ except for the factor 2 in the first equation of (18) for the model in [11]. Moreover, the simulation results show that the model in [22] is stable for this example, yet it may be unstable when $f(x)$ is only convex on $\mathcal{D}$ since it is an application of the model in [12] for solving (1) with one-sided inequality constants. For example, by only replacing $3x_1 - 2x_2 = 1$ with $x_1 - 2x_2/3 = 1$ in Example 1, we obtain anther example which $f(x)$ is only convex on $\mathcal{D}$. More importantly, when applied to this example,

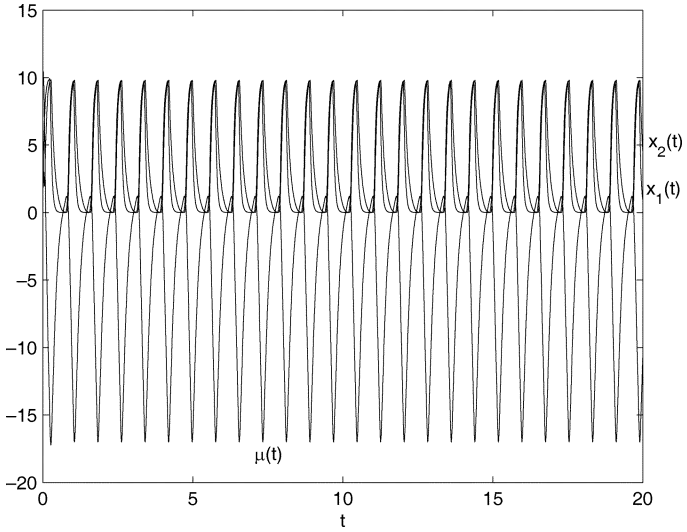


Fig. 4. Transient behavior of (18) for Example 1.

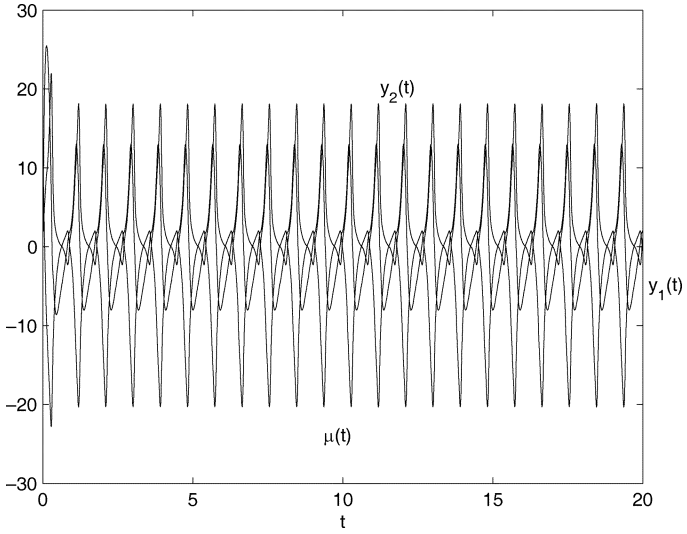


Fig. 5. Transient behavior of (19) for Example 1.

the simulation results show that the model in [22] is unstable with initial point $(1,5,10)^T$ and $\rho = 10$, while the proposed model (12)–(13) is asymptotically stable.

Next, we apply model (16)–(17) to this problem. In this case, $m = 0$, $\hat{A} = A$, $\hat{P} = D^T(DD^T)^{-1}D$, and $\hat{q} = -(I_2 - \hat{P})a + \zeta D^T(DD^T)^{-1}d$ in (16)–(17). For this example, since

$$\tilde{P} = (I_2 - \hat{P})A + \zeta\hat{P} = \frac{1}{13}\begin{pmatrix} 9\zeta - 30 & 20 - 6\zeta \\ -6\zeta - 45 & 4\zeta + 30 \end{pmatrix}$$

and

$$\tilde{A} = \beta(\tilde{P}^T + \tilde{P}) - A(I_2 - \hat{P})A - \zeta^2\hat{P} = \frac{1}{13}\begin{pmatrix} \hat{a}_{11} & \hat{a}_{12} \\ \hat{a}_{12} & \hat{a}_{22} \end{pmatrix}$$

where $\hat{a}_{11} = 18\beta\zeta - 60\beta - 225 - 9\zeta^2$, $\hat{a}_{12} = 150 - 25\beta - 12\beta\zeta + 6\zeta^2$, and $\hat{a}_{22} = 8\beta\zeta + 60\beta - 100 - 4\zeta^2$, $\tilde{P}$ is singular and $\tilde{A}$ is indefinite for any ζ and β. Then, there is no $\zeta > 0$ and $\beta \geq \zeta$ such that $\tilde{P}$ is nonsingular and $\tilde{A}$ is positive semidefinite (see [28, Th. 1]). Thus, the stability of (16) cannot be guaranteed and the output variables (17) is not well defined.

Example 2 shows that the models in [38] and [46] are unstable when A is positive semidefinite.

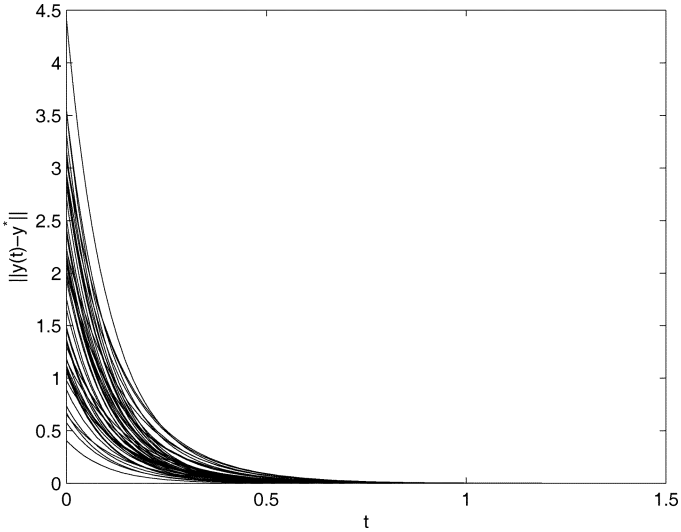


Fig. 6. Convergence behavior of the error $\|y(t) - y^*\|$ based on (29) with 50 random initial points for Example 2.

Example 2: Consider problem (1) with $\mathcal{X} = \{x \in R^4 | -10 \leq x_i \leq 10 \text{ for } i = 1,2,3,4\}$, $D = (I_2, I_2)$, $d = e_2$, $a = (-1,1,-1,1)^T$, $E = (e_2, e_2)$ and

$$A = \begin{pmatrix} E & 0 \\ 0 & E \end{pmatrix}.$$

Then, its optimal solutions are $x^* = (x_1^*, 1 - x_1^*, 1 - x_1^*, x_1^*)^T$ $(-9 \leq x_1^* \leq 10)$.

Since A is positive semidefinite, the proposed model (12)–(13) can be used to solve this problem from the analysis in Section III. In this case, (12)–(13) can be simplified as (29)–(30) with $\alpha = 1$. All simulation results show that (29) always converges to one of its equilibrium point $y^* = x^*$. For example, Fig. 6 depicts the convergence behavior of the error $\|y(t) - y^*\|$ based on (29) with 50 random initial points and $\rho = 10$.

To make a comparison, we first solve this problem using (14), which can be written as

$$\begin{cases} \dfrac{dx_i}{dt} = \rho(\min\{10, \max\{\mu_1 - x_{i+1} + 1, -10\}\} - x_i), & i = 1,3 \\ \dfrac{dx_i}{dt} = \rho(\min\{10, \max\{\mu_2 - x_{i-1} - 1, -10\}\} - x_i), & i = 2,4 \\ \dfrac{d\mu_i}{dt} = \rho(1 - x_i - x_{i+2}), \qquad i = 1,2. \end{cases} \tag{31}$$

Then, one can easily verify that $x(t) = [x_1(t), 1 - x_1(t), x_1(t), 1 - x_1(t), \mu_1(t), 2 - \mu_1(t)]^T$ is a solution of (31), where $x_1(t) = 0.5 + \sqrt{2}\gamma\sin(\sqrt{2}\rho t)/4$ and $\mu_1(t) = \gamma\cos(\sqrt{2}\rho t)/2$ with $|\gamma| \leq 19(2 - \sqrt{2})$. Obviously, $\forall \rho > 0$, this solution is unstable whenever $\gamma \neq 0$. Thus, the model in [6] and [38] cannot be used to solve this problem.

Next, when applied to this problem, (15) becomes

$$\begin{cases} \dfrac{dy_i}{dt} = \rho(\mu_1 - y_i - x_{i+1} + 1), \qquad i = 1,3 \\ \dfrac{dy_i}{dt} = \rho(\mu_2 - x_{i-1} - y_i - 1), \qquad i = 2,4 \\ \dfrac{d\mu_i}{dt} = \rho(1 - x_i - x_{i+2}), \qquad i = 1,2, \end{cases} \tag{32}$$

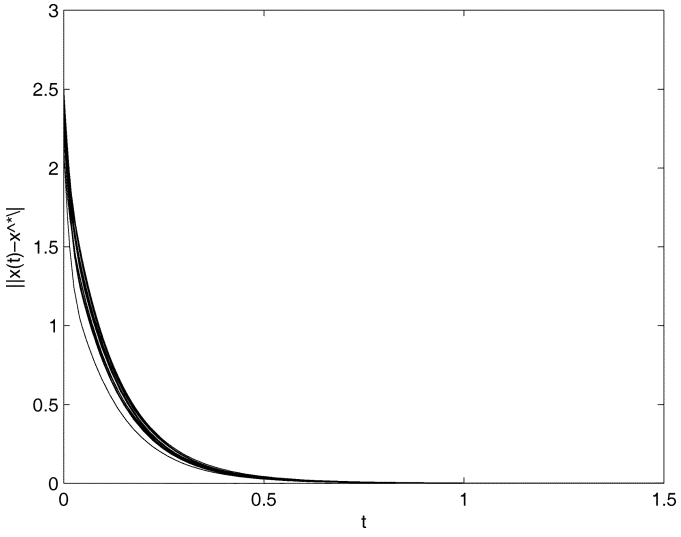


Fig. 7. Convergence behavior of the error $\|x(t)-x^*\|$ based on (12)–(13) with 20 random initial points for Example 3.

TABLE II
NUMERICAL RESULTS OF EXAMPLE 3 WITH $\rho = 10$

Model	initial point	t_f	Iter.	CPU (sec.)	$\|x(t_f)-x^*\|$
(12)-(13)	$8e_{27}$	2.209726	44	0.125	2.936978×10^{-6}
	$-e_{27}$	1.883664	49	0.140625	2.955204×10^{-6}
(14)	$8e_{44}$	16.476723	1882	10.4375	5.110288×10^{-7}
	$-e_{44}$	16.190693	1857	10.15625	6.155937×10^{-7}
(15)	$8e_{44}$	162.393603	2150	11.3125	1.198029×10^{-5}
	$-e_{44}$	165.118269	2021	11.25	9.536314×10^{-6}
(18)	$8e_{44}$	1.718624	64	0.359375	4.167837×10^{-7}
	$-e_{44}$	1.640963	68	0.375	1.383116×10^{-6}
(19)	$8e_{44}$	1.696539	63	0.359375	1.442563×10^{-7}
	$-e_{44}$	1.513001	66	0.375	7.384777×10^{-7}
(20)-(21)	$8e_{27}$	3.26719	93	0.296875	2.282522×10^{-7}
	$-e_{27}$	3.68074	84	0.265625	2.218286×10^{-7}

where $x = P_\Omega(y)$. We can see that a solution $y(t) = [y_1(t), 1-y_1(t), y_1(t), 1-y_1(t), \mu_1(t), 2-\mu_1(t)]^T$ of (32) is unstable whenever $\gamma \neq 0$, where $y_1(t) = 0.5 - \sqrt{2}\gamma\cos(\sqrt{2}\rho t)/4$ and $\mu_1(t) = \gamma\sin(\sqrt{2}\rho t)/2$ with $|\gamma| \leq 19\sqrt{2}$. Thus, the model in [39] and [46] cannot be used to solve this problem.

Example 3: To show the effectiveness of the proposed model (12)–(13) for convex quadratic programming problem, we consider

$$\begin{cases} \min & f(x) = \frac{1}{2}\sum_{k=0}^{4}(x_{3k+1} - 2x_{3k+2} + x_{3k+3})^2 \\ \text{s.t.} & 3x_{3k+1} + x_{3k+2} + 2x_{3k+3} = 60, \quad k = 0,1,2,3,4, \\ & \left.\begin{array}{l} -7 \leq x_{3k+1} - x_{3k-2} \leq 6 \\ -7 \leq x_{3k+2} - x_{3k-1} \leq 7 \\ -7 \leq x_{3k+3} - x_{3k} \leq 6 \end{array}\right\}, \quad k = 1,2,3,4, \\ & \left.\begin{array}{l} 8 \leq x_{3k+1} \leq 20 \\ 6 \leq x_{3k+2} \leq 30 \\ 3 \leq x_{3k+3} \leq 16 \end{array}\right\}, \quad k = 0,1,2,3,4. \end{cases}$$

Then, its optimal solution is x^* with $x^*_{3k+1} = 60 - 5x^*_{3k+2}$ and $x^*_{3k+3} = 7x^*_{3k+2} - 60$ $(9 \leq x^*_{3k+2} \leq 10.4)$ for $k = 0,1,2,3,4$.

Obviously the proposed model (12)–(13) with $\alpha = 1$ can be applied to solve this problem since $f(x)$ is convex on R^{15}. All simulation results show that (12) always converges to an equilibrium point (y^*, z^*) and the output variable always converges to an optimal solution of this problem. For example, Fig. 7 depicts the convergence behavior of the error $\|x(t) - x^*\|$ based on (12)–(13) with 20 random initial points and $\rho = 10$.

Next, we compare the proposed model (12) with models (14), (15), and (18)–(20). Table II reports the numerical results with various initial points obtained by these models, respectively, where Iter. represents the iterative number in ODE23S and t_f denotes the time that the stopping criterion, the norm of the right-side hand of these models being less than 10^{-5}, is met. From Table II, we can see that the proposed model has a faster convergence than other models.

Example 4: To show the performance of the proposed neural network (12)–(13) for linear programming, we consider the following linear assignment problem (see [27] and [36]):

$$\begin{cases} \min & \sum_{i=1}^{n}\sum_{i=1}^{n} w_{ij}x_{ij} \\ \text{s.t.} & \sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1,2,\ldots,n \\ & \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1,2,\ldots,n \\ & x_{ij} \in \{0,1\}, \qquad i,j = 1,2,\ldots,n. \end{cases} \tag{33}$$

According to [27] and [36], if this problem has a unique optimal solution, then it is equivalent to a linear programming problem

$$\begin{cases} \min & f(x) = a^T x \\ \text{s.t.} & Dx = d, \quad x \in \mathcal{X} \end{cases}$$

where $x = \text{vec}(X) = (x_{11}, x_{12}, \ldots, x_{1n}, \ldots, x_{n1}, x_{n2}, \ldots, x_{nn})^T$, $\mathcal{X} = \{x \in R^{n^2} \mid 0 \leq x_{ij} \leq 1 \text{ for } i,j = 1,2,\ldots,n\}$, $a = vec(W)$, $d = e_{2n-1}$, $M = (I_{n-1}, 0)_{(n-1)\times n}$, and

$$D = \begin{pmatrix} e_n^T & 0 & \cdots & 0 \\ 0 & e_n^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ M & M & \cdots & M \end{pmatrix}_{(2n-1)\times n^2}.$$

Thus, problem (33) can be solved by the proposed neural network (29)–(30) with $C = I_{n^2} - D^T(DD^T)^{-1}D$ and $q = D^T(DD^T)^{-1}d - Ca$ when it has a unique optimal solution.

As an example, we let $a = \text{vec}(W) = (3,2,4,1, 2,1.5,2.5,3,4,3.5,3,1.5,3,2,1,2)^T$, then (33) has a unique solution $x^* = (0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0)^T$. When applied to this problem, all simulation results show that (29) always converges to one of its equilibrium points and the output variables of (30) are asymptotically stable at x^*. For example, Fig. 8 depicts the convergence behavior of the error $\|x(t)-x^*\|$ based on (29)–(30) with 20 random initial points and $\rho = 10$.

Compared with existing one-layer neural network for linear programming in [27], the proposed model (12)–(13) has no adjustable parameter, and its right-hand side is continuous. Thus, it avoids the difficulty of choosing network parameter.

From the above examples and their simulation results, we have the following remark.

Remark 5:

i) When applied to Example 1, the simulation results show that all models in [11], [14], [34], [37], [38], and [46] with initial point $(1,5,10)^T$ and $\rho = 10$ are unstable.

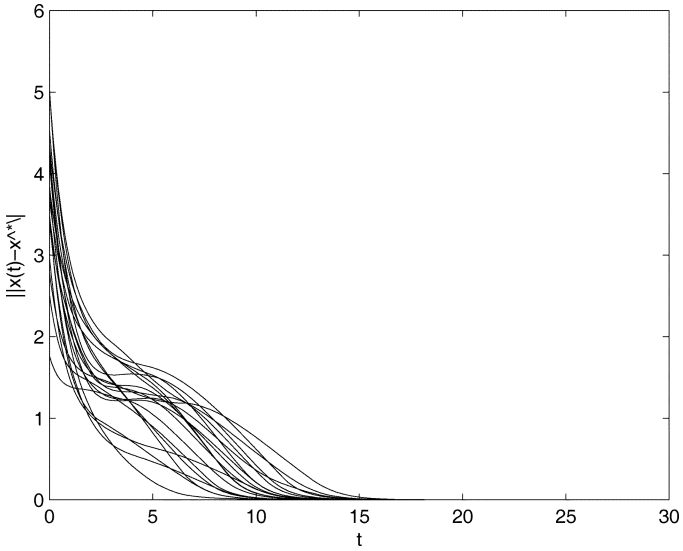


Fig. 8. Convergence behavior of the error $\|x(t)-x^*\|$ based on (29)–(30) with 20 random initial points for Example 4.

ii) For models (14), (15), (18), and (19) and the model in [22], the simulation results show that they are stable for Example 3, yet unlike model (12), their stability and convergence might not be guaranteed. In particular, (14) and (15) are unstable even when A is positive semidefinite (see Example 2), and the model in [22] and (18) and (19) are also unstable when $f(x)$ is only convex on $\mathcal{D}$ (see Example 1).

iii) Based on the similar analysis, we can conclude that the model in [28] could not be used to solve Examples 2 and 3 since there is no parameter ζ such that its output equation can be well defined.

V. Conclusion

In this paper, we have presented a one-layer neural network for solving linear and quadratic programming problems in real time by introducing some new vectors. The stability and the convergence of the proposed neural network are proved under the condition that the objective function is convex on the set defined by equality constraints. Compared with existing one-layer neural networks for quadratic programming problems, the proposed neural network has least neurons and requires weak stability conditions. Our simulation results on four examples clearly demonstrate the convergence behavior and the attractiveness of the proposed neural network.

Acknowledgment

The authors would like to thank the Associate Editor and three anonymous referees for their valuable comments and suggestions on earlier versions of this paper.

References

[1] M. Avriel, *Nonlinear Programming: Analysis and Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1976.

[2] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming-Theory and Algorithms*, 2nd ed. New York: Wiley, 1993.

[3] A. Bouzerdorm and T. R. Pattison, "Neural network for quadratic optimization with bound constraints," *IEEE Trans. Neural Netw.*, vol. 4, no. 2, pp. 293–304, Mar. 1993.

[4] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*. New York: Wiley, 1993.

[5] R. Fletcher, *Practical Methods of Optimization*. New York: Wiley, 1981.

[6] T. L. Friesz, D. H. Bernstein, N. J. Mehta, R. L. Tobin, and S. Ganjlizadeh, "Day-to-day dynamic network disequilibria and idealized traveler information systems," *Oper. Res.*, vol. 42, pp. 1120–1136, 1994.

[7] M. Forti, P. Nistri, and M. Quincampoix, "Generalized neural network for nonsmooth nonlinear programming problems," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 51, no. 9, pp. 1741–1754, Sep. 2004.

[8] X. B. Gao, "A neural network for a class of extended linear variational inequalities," *Chin. J. Electron.*, vol. 10, no. 4, pp. 471–475, 2001.

[9] X. B. Gao, "A novel neural network for nonlinear convex programming," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 613–621, May 2004.

[10] X. B. Gao and L. L. Du, "A neural network with finite-time convergence for a class of variational inequalities," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2006, vol. 4113, pp. 32–41.

[11] X. B. Gao and L.-Z. Liao, "A neural network for monotone variational inequalities with linear constraints," *Phys. Lett. A*, vol. 307, no. 2-3, pp. 118–128, 2003.

[12] X. B. Gao and L.-Z. Liao, "A novel neural network for a class of convex quadratic minimax problems," *Neural Comput.*, vol. 18, no. 8, pp. 1818–1846, 2006.

[13] X. B. Gao and L.-Z. Liao, "A new projection-based neural network for constrained variational inequalities," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, pp. 622–628, Mar. 2009.

[14] H. Ghasabi-Oskoei and N. Mahdavi-Amiri, "An efficient simplified neural network for solving linear and quadratic programming problems," *Appl. Math. Comput.*, vol. 175, no. 1, pp. 452–464, 2006.

[15] M. P. Glazos, S. Hui, and S. H. Żak, "Sliding modes in solving convex programming problems," *SIAM J. Control Optim.*, vol. 36, no. 2, pp. 680–697, 1998.

[16] Q. M. Han, L.-Z. Liao, H. D. Qi, and L. Q. Qi, "Stability analysis of gradient-based neural networks for optimization problem," *J. Global Optim.*, vol. 19, no. 1, pp. 363–381, 2001.

[17] P. T. Harker and J. S. Pang, "Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms, and applications," *Math. Progr.*, ser. B, vol. 48, pp. 161–220, 1990.

[18] B. S. He and L.-Z. Liao, "Improvements of some projection methods for monotone nonlinear variational inequalities," *J. Optim. Theory Appl.*, vol. 112, no. 1, pp. 111–128, 2002.

[19] J. J. Hopield and D. W. Tank, "Computing with neural circuits: A model," *Science*, vol. 233, no. 4764, pp. 625–633, 1986.

[20] X. L. Hu and J. Wang, "Design of general projection neural networks for monotone linear variational inequalities and linear and quadratic optimization problems," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 37, no. 5, pp. 1414–1421, Oct. 2007.

[21] X. L. Hu and J. Wang, "An improved dual neural network for solving a class of quadratic programming problems and its k-winners-take-all application," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 2022–2031, Dec. 2008.

[22] X. L. Hu and B. Zhang, "A new recurrent neural network for solving convex quadratic programming problems with an application to the k-winners-take-all problem," *IEEE Trans. Neural Netw.*, vol. 20, no. 4, pp. 654–664, Apr. 2009.

[23] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. CAS-35, no. 5, pp. 554–562, May 1988.

[24] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and their Applications*. New York: Academic, 1980.

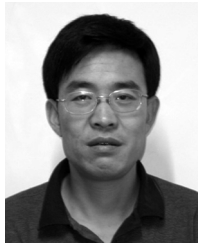
[25] Q. S. Liu and J. D. Cao, "Solving quadratic programming problems by delayed projection neural network," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1630–1634, Nov. 2006.

[26] Q. S. Liu and J. Wang, "A simplified dual neural network for quadratic programming with its KWTA application," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1500–1510, Nov. 2006.

[27] Q. S. Liu and J. Wang, "A one-layer recurrent neural network with a discontinuous activation function for linear programming," *Neural Comput.*, vol. 20, no. 5, pp. 1366–1383, 2008.

[28] Q. S. Liu and J. Wang, "A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming," *IEEE Trans. Neural Netw.*, vol. 19, no. 4, pp. 558–570, Apr. 2008.

[29] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equation in Several Variables*. New York: Academic, 1970.
[30] A. Rodríguez-Vázquez, R. Domínguez-Castro, J. L. Huertas, and E. Sánchez-Sinencio, "Nonlinear switched-capacitor 'neural networks' for optimization problems," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 384–397, Mar. 1990.
[31] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice-Hall, 1991.
[32] M. V. Solodov and P. Tseng, "Modified projection-type methods for monotone variational inequalities," *SIAM J. Control Optim.*, vol. 34, no. 5, pp. 1814–830, 1996.
[33] D. W. Tank and J. J. Hopield, "Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, no. 5, pp. 533–541, May 1986.
[34] Q. Tao, J. Cao, and D. Sun, "A simple and high performance neural network for quadratic programming problems," *Appl. Math. Comput.*, vol. 124, no. 2, pp. 251–260, 2001.
[35] P. Tseng, "A modified forward-backward splitting method for maximal monotone mappings," *SIAM J. Control Optim.*, vol. 38, no. 2, pp. 431–446, 2000.
[36] J. Wang, "Primal and dual assignment network," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 784–790, May 1997.
[37] Y. S. Xia, "A new neural network for solving linear programming and quadratic programming problems," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1544–1547, Nov. 1996.
[38] Y. S. Xia, "An extended projection neural network for constrained optimization," *Neural Comput.*, vol. 16, no. 4, pp. 863–883, 2004.
[39] Y. S. Xia and G. Feng, "A new neural network for solving nonlinear projected equations," *Neural Netw.*, vol. 20, no. 5, pp. 577–589, 2007.
[40] Y. S. Xia, G. Feng, and J. Wang, "A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equations," *Neural Netw.*, vol. 17, no. 7, pp. 1003–1015, 2004.
[41] Y. S. Xia, G. Feng, and J. Wang, "A novel recurrent neural network for solving nonlinear optimization problems with inequality constraints," *IEEE Trans. Neural Netw.*, vol. 19, no. 8, pp. 1340–1353, Aug. 2008.
[42] Y. S. Xia and J. S. Wang, "Neural network for solving linear programming problems with bounded variables," *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 515–519, Mar. 1995.
[43] Y. S. Xia and J. Wang, "A dual neural network for kinematic control of redundant robot manipulators," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 31, no. 1, pp. 147–151, Feb. 2001.
[44] Y. S. Xia and J. Wang, "A general projection neural network for solving monotone variational inequalities and related optimization problems," *IEEE Trans. Neural Netw.*, vol. 15, no. 2, pp. 318–328, Mar. 2004.
[45] Y. S. Xia and J. Wang, "A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 7, pp. 1385–1394, Jul. 2004.
[46] Y. S. Xia and J. Wang, "Solving variational inequality problems with linear constraints based on a novel neural network," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2007, vol. 4493, pp. 95–104.
[47] Y. Zhang and J. Wang, "A dual neural network for convex quadratic problem subject to linear equality and inequality constraints," *Phys. Lett. A*, vol. 298, no. 4, pp. 271–278, 2002.



Xingbao Gao received the B.S. and M.S. degrees in mathematics from Shaanxi Normal University, Xi'an, China, in 1988 and 1991, respectively, and the Ph.D. degree in applied mathematics from Xidian University, Xi'an, China, in 2000.

Currently, he is a Professor at the College of Mathematics and Information Science, Shaanxi Normal University. His research interests include optimization theory, algorithms and its applications, neural networks, and numerical method for partial differential equations.

Li-Zhi Liao received the B.S. degree in applied mathematics from Tsinghua University, Beijing, China, in 1984, and the M.S. and Ph.D. degrees at the School of Operations Research and Industrial Engineering from Cornell University, Ithaca, NY, in 1988 and 1990, respectively.

Currently, he is a Professor at the Mathematics Department, Hong Kong Baptist University, Hong Kong. His recent research interests include optimization, optimal control, variational inequality, and scientific computing.