# Design of General Projection Neural Networks for Solving Monotone Linear Variational Inequalities and Linear and Quadratic Optimization Problems

Xiaolin Hu and Jun Wang

***Abstract*—Most existing neural networks for solving linear variational inequalities (LVIs) with the mapping $Mx + p$ require positive definiteness (or positive semidefiniteness) of $M$. In this correspondence, it is revealed that this condition is sufficient but not necessary for an LVI being strictly monotone (or monotone) on its constrained set where equality constraints are present. Then, it is proposed to reformulate monotone LVIs with equality constraints into LVIs with inequality constraints only, which are then possible to be solved by using some existing neural networks. *General projection neural networks* are designed in this correspondence for solving the transformed LVIs. Compared with existing neural networks, the designed neural networks feature lower model complexity. Moreover, the neural networks are guaranteed to be globally convergent to solutions of the LVI under the condition that the linear mapping $Mx + p$ is monotone on the constrained set. Because quadratic and linear programming problems are special cases of LVI in terms of solutions, the designed neural networks can solve them efficiently as well. In addition, it is discovered that the designed neural network in a specific case turns out to be the primal-dual network for solving quadratic or linear programming problems. The effectiveness of the neural networks is illustrated by several numerical examples.**

***Index Terms*—Global convergence, linear programming, linear variational inequality (LVI), quadratic programming, recurrent neural network.**

## I. Introduction

In this correspondence, we are concerned with solving the following linear variational inequality (LVI): find $x^* \in \Omega$ such that

$$(Mx^* + p)^T(x - x^*) \geq 0 \qquad \forall x \in \Omega \tag{1}$$

where $M \in \Re^{n\times n}$, and $p \in \Re^n$, and

$$\Omega = \{x \in \Re^n | Ax \in \mathcal{Y}, Bx = c, x \in \mathcal{X}\} \tag{2}$$

with $A \in \Re^{m\times n}$, $B \in \Re^{r\times n}$, and $c \in \Re^r$. $\mathcal{X}$ and $\mathcal{Y}$ are two box sets defined as $\mathcal{X} = \{x \in \Re^n | \underline{x} \leq x \leq \overline{x}\}$, $\mathcal{Y} = \{y \in \Re^m | \underline{y} \leq y \leq \overline{y}\}$, where $\underline{x}, \overline{x} \in \Re^n$, and $\underline{y}, \overline{y} \in \Re^m$. Note that any component of $\underline{x}, \underline{y}$ may be set to $-\infty$ and any component of $\overline{x}, \overline{y}$ may be set to $\infty$. Without loss of generality, we assume that $\underline{y} < \overline{y}$, since if $\underline{y}_i = \overline{y}_i$ for some $i$, then the corresponding inequality constraints can be incorporated into $Bx = c$.

The above LVI is closely related to the following quadratic programming problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}x^T Mx + p^T x \\ \text{subject to} \quad & x \in \Omega \end{aligned} \tag{3}$$

Manuscript received November 25, 2006; revised April 11, 2007. This work was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project CUHK4165/03E. This paper was recommended by Associate Editor M. Qiao.

The authors are with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: jwang@acae.cuhk.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2007.903706

where the parameters are the same as in (1). It is well known that (e.g., [1] and [2]) if $M$ is symmetric and positive semidefinite, the two problems are actually equivalent. If $M = 0$, the above problem degenerates to a linear programming problem.

The optimization problem (3) has a wide variety of scientific and engineering applications, e.g., regression analysis, image and signal processing, parameter estimation, robot control, to list a few. This fact partly addresses the significance of the study of LVI (1). In addition, LVI has many applications other than optimization, including analysis of piecewise-linear resistive circuits, bimatrix equilibrium points problem, economic equilibrium modeling, traffic network equilibrium modeling and structural analysis, and so on; see [1] and [2] for a comprehensive view of LVI and nonlinear variational inequality (NVI). In past years, numerous numerical methods have been proposed for solving LVIs and NVIs, e.g., see [1]–[3] and references therein. Another branch of methods refers to recurrent neural networks, which are of particular interest in real-time applications and parallel computation [4], [5]. For example, in [6], a neural network approach for solving LVI with bound (box or sphere) constraints was proposed. In [7], another neural network model was devised for solving LVI with bound constraints. In [8]–[11] and [25], a projection neural network for solving NVI and related optimization problems with bound constraints was developed. More recently, two neural networks capable of solving monotone NVI with general constraints were invented in [13]–[15]. Because optimization problems are special cases of variational inequalities, the aforementioned neural networks can be used to solve some optimization problems as well (usually convex problems). In addition, many neural networks were developed for optimization (e.g., [4], [5], [16]–[23], and references therein), and some of them will be discussed briefly in Section III-C for solving (3).

In the design of recurrent neural networks for solving variational inequalities and related optimization problems, a critical issue is stability conditions. In this correspondence, an LVI (1) is called a monotone (strictly monotone) LVI if the mapping $Mx + p$ is monotone (strictly monotone) on the constrained set $\Omega$, and one of our primary purposes is to solve such monotone LVIs. Most of the existing neural networks mentioned above that are capable of solving (1) or (3), e.g., those in [13], [19], [20], and [22], require that $Mx + p$ is monotone (strictly monotone) on $\Re^n$, or equivalently, $M$ is positive semidefinite (positive definite). Because of the presence of equality constraints in the constrained set $\Omega$ in (2), the positive semidefiniteness (or positive definiteness) of $M$ is not a necessary condition for ensuring the monotonicity (or strictly monotonicity) of $Mx + p$ on $\Omega$, as will be made clearly in Section II. A natural question arises, i.e., can these existing neural networks be applied directly to solve an LVI (1) that is (strictly) monotone on $\Omega$ instead of on $\Re^n$? Unfortunately, the answer is no, as shown by two examples in Sections III and III-D. This fact motivates us to reformulate the problem in order to apply existing neural network models or design new models for solving it. Another critical issue in designing recurrent neural networks for engineering applications is model complexity. We aim at designing neural networks of lower complexity than existing ones for solving monotone LVIs.

The rest of this correspondence is organized as follows. In Section II, some preliminaries are introduced. The relationship between the monotonicity of a linear mapping $Mx + p$ and the positive semidefiniteness of $M$ is revealed. In Section III, a general projection neural network (GPNN) [24] is designed to solve LVI (1) with hybrid linear constraints. Some special cases of (1) and the corresponding GPNNs are also discussed in this section. In Section IV, two numerical examples are presented. Finally, Section V concludes this correspondence.

## II. Problem Formulation

For the convenience of discussion, it is necessary to introduce some notations and definitions first. Throughout this correspondence, the following notations are used. The two capitals $I$ and $O$ stand for the identity matrix and zero matrix, respectively, and their dimensions can be determined from the context. For simplicity, a square matrix $M \geq 0$ means that it is positive semidefinite, and $M > 0$ means that it is positive definite. Denote $\Omega^*$ as the solution set of (1). Throughout this correspondence, it is assumed that $\Omega^* \neq \emptyset$. Moreover, we assume $\text{Rank}(B) = r$ in (2), which is always true for a well-posed problem.

*Definition 1:* A mapping $F : \Re^n \to \Re^n$ is said to be monotone on a set $\mathcal{K}$ if $\forall x, y \in \mathcal{K}$

$$(F(x) - F(y))^T (x - y) \geq 0.$$

$F$ is said to be strictly monotone on $\mathcal{K}$ if the strict inequality above holds whenever $x \neq y$, and strongly monotone on $\mathcal{K}$ if there exists a constant $\gamma > 0$ such that $\forall x, y \in \mathcal{K}$

$$(F(x) - F(y))^T (y - x) \geq \gamma \|x - y\|^2.$$

For a linear mapping, strict monotonicity is identical with strong monotonicity. To solve the LVI defined in (1), most of the existing neural networks such as in [13], [19], [20], and [22] require that $M \geq 0$ or $M > 0$. According to Definition 1 and the results in [2, pp. 155–156], the two conditions are necessary and sufficient for ensuring that the mapping $F(x) = Mx + p$ is monotone and strictly monotone on $\Omega$, respectively, if $\Omega$ does not include the equality constraints $Bx = c$. However, if the equality constraints are present in $\Omega$, the latter condition is weaker than the former, but a necessary and sufficient condition similar to the former can be found for the latter [see statement 3) in Lemma 1]. Before this lemma is presented, some notations need to be introduced. Since $\text{Rank}(B) = r$, without loss of generality, $B$ can be partitioned as $[B_I, B_{II}]$, where $B_I \in \Re^{r \times r}$, $B_{II} \in \Re^{r \times (n-r)}$, and $\det(B_I) \neq 0$. Then, $Bx = c$ can be decomposed into

$$(B_I, B_{II}) \begin{pmatrix} x_I \\ x_{II} \end{pmatrix} = c$$

where $x_I \in \Re^r$, and $x_{II} \in \Re^{n-r}$, which yields $x_I = -B_I^{-1} B_{II} x_{II} + B_I^{-1} c$, and

$$x = Q x_{II} + q \tag{4}$$

where

$$Q = \begin{pmatrix} -B_I^{-1} B_{II} \\ I \end{pmatrix} \in \Re^{n \times (n-r)} \quad q = \begin{pmatrix} B_I^{-1} c \\ O \end{pmatrix} \in \Re^n. \tag{5}$$

*Lemma 1:* The following statements are equivalent.

1) The mapping $Mx + p$ in (1) is monotone (strictly monotone) on $\Omega$ defined in (2).
2) $Q^T M Q \geq (>) 0$, where $Q$ is defined in (5).
3) $M$ is positive semidefinite (positive definite) on the cone $\mathcal{C} = \{\eta \in \Re^n | B\eta = 0, \eta \neq 0\}$, i.e., $\eta^T M \eta \geq (>) 0 \ \forall \eta \in \mathcal{C}$.

*Proof:* We only prove the first part of each statement while the other part can be reasoned similarly.

(i)⇔(ii): By Definition 1, the monotonicity of $Mx + p$ on $\Omega$ is equivalent to

$$(Mx - My)^T (x - y) \geq 0 \qquad \forall x, y \in \Omega.$$

As both $x, y$ satisfy $Bx = c$, they can be expressed in terms of $x_{II}$ and $y_{II}$ in the form of (4). Then, the above inequality is equivalent to

$$(x_{II} - y_{II})^T Q^T M Q (x_{II} - y_{II}) \geq 0 \qquad \forall x_{II}, y_{II} \in \bar{\Omega}$$

where

$$\bar{\Omega} = \left\{ x_{II} \in \Re^{n-r} | A(Q x_{II} + q) \in \mathcal{Y}, Q x_{II} + q \in \mathcal{X} \right\} \tag{6}$$

which is further equivalent to $Q^T M Q \geq 0$ by considering that there is no equality constraint inside $\bar{\Omega}$.

(ii)⇔(iii): The cone $\mathcal{C}$ in statement 3) can be equivalently written as $\mathcal{C} = \{Q\xi | \xi \in \Re^{n-r}, \xi \neq 0\}$, where $Q$ is defined in (5). Then, $\eta^T M \eta \geq 0$ for any $\eta \in \mathcal{C}$ is identical with $\xi^T Q^T M Q \xi \geq 0$ for any $\xi \in \Re^{n-r}$ and $\xi \neq 0$, namely $Q^T M Q \geq 0$. ■

Regarding statement 3) in Lemma 1, one thing needs to be clarified. Throughout this correspondence, when we say a matrix $M$ is positive (semi) definite without specifying any set $\mathcal{K}$, it means $\mathcal{K} = \Re^n$. In addition, regarding this statement, we have the following result.

*Theorem 1:* $M$ is positive definite on the cone $\mathcal{C}$ defined in Lemma 1 if and only if there exists $\bar{\rho} > 0$ such that for $\rho \geq \bar{\rho}$, $M + \rho B^T B > 0$.

*Proof:* If $M + \rho B^T B > 0$, then $x^T M x + \rho x^T B^T B x > 0$ $\forall x \in \Re^n$. It follows that $x^T M x > 0$ for all $x \in \Re^n, Bx = 0$. This proves the "if" part. The "only if" part can be reasoned in the same manner as analyzing the lemma in [2, Lemma 1.25, p. 68] by considering that $B^T B \geq 0$. ■

This theorem can inspire an approach for solving LVI (1) with strict monotonicity of $Mx + p$ on $\Omega$ in view of the equivalence between (1) and

$$\left( (M + \rho B^T B) x^* + p - \rho B^T c \right)^T (x - x^*) \geq 0 \qquad \forall x \in \Omega \tag{7}$$

where $\Omega$ is defined in (2). Then, many neural networks in the literature can be applied to solve this problem. One limitation of this new formulation is the difficulty in choosing an appropriate $\rho$—in doing so, one has to choose by trial-and-error which is often time consuming. Another limitation lies in that the formulation takes effect only for strictly monotone LVIs on $\Omega$. In the following, we derive another equivalent LVI to (1) via variable substitution.

A simple technique in solving mathematical problems with equalities is to express some unknowns in terms of the others and then substitute them into the problem, which will often reduce the size of the original problem. Such a technique can be adopted as well in solving LVIs. Based on $Bx = c$, we have obtained $x = Q x_{II} + q$ in (4). Since $x^*$ in (1) should satisfy $Bx = c$ too, we have $x^* = Q x_{II}^* + q$. Substitution of $x$ and $x^*$ into (1) gives

$$(MQx_{II}^* + Mq + p)^T Q (x_{II} - x_{II}^*) \geq 0 \qquad \forall Q x_{II} + q \in \Omega'$$

where $\Omega'$ is the set $\Omega$ in (2) with $Bx = c$ deleted, or

$$\left( Q^T M Q x_{II}^* + Q^T M q + Q^T p \right)^T (x_{II} - x_{II}^*) \geq 0 \qquad \forall x_{II} \in \bar{\Omega}$$

where $\bar{\Omega}$ is defined in (6). Now, the original LVI is equivalently converted into another LVI. Let

$$u = x_{II} \quad \bar{M} = Q^T M Q \quad \bar{p} = Q^T M q + Q^T p$$
$$\bar{A} = \begin{pmatrix} AQ \\ -B_I^{-1} B_{II} \end{pmatrix}$$
$$\mathcal{V} = \left\{ v \in \Re^{m+r} \,\middle|\, \begin{pmatrix} \underline{y} - Aq \\ \underline{x}_I - B_I^{-1} c \end{pmatrix} \leq v \leq \begin{pmatrix} \overline{y} - Aq \\ \overline{x}_I - B_I^{-1} c \end{pmatrix} \right\}$$
$$\mathcal{U} = \{ u \in \Re^{n-r} | \underline{x}_{II} \leq u \leq \overline{x}_{II} \}$$

where $\underline{x}_I = (\underline{x}_1, \ldots, \underline{x}_r)^T$, $\underline{x}_{II} = (\underline{x}_{r+1}, \ldots, \underline{x}_n)^T$, $\overline{x}_I = (\overline{x}_1, \ldots, \overline{x}_r)^T$, and $\overline{x}_{II} = (\overline{x}_{r+1}, \ldots, \overline{x}_n)^T$. Then the new LVI can be rewritten in the following compact form:

$$(\bar{M} u^* + \bar{p})^T (u - u^*) \geq 0 \qquad \forall u \in \bar{\Omega} \tag{8}$$

where

$$\bar{\Omega} = \{ u \in \Re^{n-r} | \bar{A} u \in \mathcal{V}, u \in \mathcal{U} \}.$$

Compared with the original LVI in (1) and the LVI in (7), the new formulation (8) has fewer variables while the equality constraints are absorbed. Again, existing neural networks may be applied to solve this new problem under the condition that $\bar{M}$ is positive definite or positive semidefinite, i.e., $Mx + p$ is strictly monotone or monotone on $\Omega$ according to Lemma 1. However, those neural networks will have more neurons or states than what we will design in Section III-A, and this point will be made clearly in Section III-C.

## III. Main Results

In [24], the GPNN was presented for solving general variational inequalities with the box constraints $\mathcal{X}$ defined in (2), i.e.,

$$\frac{dx}{dt} = \lambda D \{ -G(x) + P_{\mathcal{X}} (G(x) - F(x)) \}$$

where $F, G : \Re^n \to \Re^n$, $D \in \Re^{n \times n}$, and $\lambda > 0$ is a scalar; and $P_{\mathcal{X}} : \Re^n \to \mathcal{X}$ is a projection operator defined as $(P_{\mathcal{X}}(x_1), \ldots, P_{\mathcal{X}}(x_n))^T$ with

$$P_{\mathcal{X}}(x_i) = \begin{cases} \underline{x}_i, & x_i < \underline{x}_i \\ x_i, & \underline{x}_i \leqslant x_i \leqslant \overline{x}_i \\ \overline{x}_i, & x_i > \overline{x}_i. \end{cases} \tag{9}$$

This operator is called the "activation function." In [25], a linear case of this neural network (i.e., both $F$ and $G$ are linear mappings) is further studied. In this section, we will design a linear case of the neural network with low complexity and high performance for solving LVI (8) and consequently the original LVI (1). To attain this goal, some techniques in optimization and stability analysis will be focused.

### A. Model Design

First of all, a necessary and sufficient condition characterizing the solutions of LVI (8) is provided below, which will play a critical role in the design of the GPNN.

*Theorem 2:* $u^* \in \Re^{n-r}$ is a solution of (8) if and only if there exists $v^* \in \Re^{m+r}$ such that $w^* = ((u^*)^T, (v^*)^T)^T$ satisfies $\tilde{N} w^* \in \mathcal{W}$ and

$$(\tilde{M} w^* + \tilde{p})^T (w - \tilde{N} w^*) \geq 0 \qquad \forall w \in \mathcal{W} \tag{10}$$

where

$$\tilde{M} = \begin{pmatrix} \bar{M} & -\bar{A}^T \\ O & I \end{pmatrix} \quad \tilde{N} = \begin{pmatrix} I & O \\ \bar{A} & O \end{pmatrix}$$
$$\tilde{p} = \begin{pmatrix} \bar{p} \\ O \end{pmatrix} \quad \mathcal{W} = \mathcal{U} \times \mathcal{V}.$$

*Proof:* Define a function $\phi(u) = (\bar{M} u^* + \bar{p})^T (u - u^*)$ from $\Re^{n-r}$ into itself, where $u^*$ is a solution of LVI (8). Then, $u^*$ solves the following optimization problem:

$$\begin{aligned} &\text{minimize} \quad \phi(u) \\ &\text{subject to} \quad \bar{A} u \in \mathcal{V}, u \in \mathcal{U} \end{aligned} \tag{11}$$

which is a linear programming problem. Define a Lagrangian function associated with (11) as

$$L(u, v, \eta) = \phi(u) - v^T (\bar{A} u - \eta)$$

where $v \in \Re^{m+r}$ is the Lagrangian multiplier, and $\eta \in \mathcal{U}$. According to the well-known saddle point theorem [27], $u^*$ is an optimal solution of (11) if and only if there exists $v^* \in \Re^{m+r}$ and $\eta^* \in \Re^{m+r}$ such that

$$L(u^*, v, \eta^*) \leq L(u^*, v^*, \eta^*) \leq L(u, v^*, \eta)$$

for $u \in \mathcal{U}$, $v \in \Re^{m+r}$, and $\eta \in \mathcal{V}$, or explicitly

$$\begin{aligned} -v^T (\bar{A} u^* - \eta^*) \leq & -(v^*)^T (\bar{A} u^* - \eta^*) \\ & \leq (\bar{M} u^* + \bar{p})^T (u - u^*) - (v^*)^T (\bar{A} u - \eta) \end{aligned}$$

for $u \in \mathcal{U}$, $v \in \Re^{m+r}$, and $\eta \in \mathcal{V}$. The left inequality above implies $\bar{A} u^* = \eta^*$, and the right inequality implies

$$(\bar{M} u^* + \bar{p})^T (u - u^*) - (v^*)^T (\bar{A} u - \bar{A} u^*) + (v^*)^T (\eta - \eta^*) \geq 0$$

for $u \in \mathcal{U}, \eta \in \mathcal{V}$, which follows

$$(\bar{M} u^* + \bar{p} - \bar{A}^T v^*)^T (u - u^*) + (v^*)^T (\eta - \bar{A} u^*) \geq 0$$

for $u \in \mathcal{U}$, and $\eta \in \mathcal{V}$. By replacing $\eta$ with $v$, the above inequality can be written in compact form (10), which proves the "only if" part of the theorem. Conversely, if the above inequality holds, since $u^*$ is also a parameter in $\phi(u)$, according to the definition of $u^*$, it solves LVI (8). The "if" part is thus proved. ■

Inequality (10) represents a class of *general variational inequality* [24]. Thus, the following GPNN presented in [24] can be applied to solve the problem:

- State equation

$$\frac{dw}{dt} = \lambda (\tilde{M} + \tilde{N})^T \left\{ -\tilde{N} w + P_{\mathcal{W}} \left( (\tilde{N} - \tilde{M}) w - \tilde{p} \right) \right\} \tag{12a}$$

- Output equation

$$x = Qu + q \tag{12b}$$

where $\lambda > 0$ is a scalar, $Q, q$ are defined in (5), and $P_{\mathcal{W}}(\cdot)$ is a projection operator defined in (9), which is used as the activation function. Componentwise, there are totally $n + m$ scalar activation functions, which implies that there should be $n + m$ neurons in the network. In addition, for this neural network, the number of neurons is equal to the number of states $w_i$.

### B. Stability Analysis

In what follows, let $\mathcal{W}^e = \mathcal{U}^e \times \mathcal{V}^e$ denote the equilibrium set of neural network (12). According to [28], $w^* = ((u^*)^T, (v^*)^T)^T$ is a solution of (10) if and only if it is a solution of the following equation:

$$P_{\mathcal{W}}\left((\tilde{N} - \tilde{M})w - \tilde{p}\right) = \tilde{N}w.$$

It is easy to verify that when $\bar{M} \geq 0$, $\tilde{M} + \tilde{N}$ is nonsingular, which leads to the following lemma.

*Lemma 2:* If $\bar{M} \geq 0$, then $\Omega^* = \{x \in \Re^n | x = Qu + q, u \in \mathcal{U}^e\}$.

*Theorem 3:* Consider the neural network (12) for solving LVI (1) with $\Omega$ defined in (2).

1) If $\bar{M} \geq 0$, then the state of the neural network is stable in the sense of Lyapunov and globally convergent to an equilibrium, which corresponds to an exact solution of the LVI by the output equation.
2) Furthermore, if $\bar{M} > 0$, then the output $x(t)$ of the neural network is globally asymptotically stable at the unique solution $x^*$ of the LVI.

*Proof:* Part 1 of Theorem 3 directly follows from Lemma 2 and [24, Corollary 4] by noting that

$$\tilde{M}^T\tilde{N} = \begin{pmatrix} \bar{M}^T & O \\ O & O \end{pmatrix} \geq 0$$

because $\bar{M} \geq 0$ by assumption. If in addition $\bar{M} > 0$, the mapping $\bar{M}x + \bar{p}$ is then strongly monotone on $\bar{\Omega}$. According to [1, Corollary 3.2], there exists a unique solution $u^*$ to LVI (8). It follows from Lemma 2 and part 1 of the theorem that the output of the neural network $x(t)$ is globally asymptotically stable at $x^* = Qu^* + q$, i.e., the unique solution of LVI (1). ■

Note that $M \geq (>)0$ implies $\bar{M} = Q^T M Q \geq (>)0$.

### C. Model Comparisons

In this subsection, several salient recurrent neural network (RNN) are presented for solving the LVI (1) in comparison with the designed GPNN (12). The criteria include the stability conditions and the structural complexity. A typical quantity characterizing the structural complexity is the number of neurons in a network as each neuron entails an activation function. However, in many cases, this number alone cannot characterize the overall structural complexity. In what follows, the numbers of states in different RNNs are also compared.

According to Xia [13], [14], the following *extended projection neural network* can be used to solve LVI (1):

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \lambda\begin{pmatrix} -x + P_{\mathcal{X}}\left(x - Mx - p - (A')^T y + B^T z\right) \\ -y + (y + A'x - a')^+ \\ -Bx + c \end{pmatrix} \quad (13)$$

where $\lambda > 0$ is a scaling factor, and

$$A' = \begin{pmatrix} A \\ -A \end{pmatrix}, \quad a' = \begin{pmatrix} \overline{y} \\ -\underline{y} \end{pmatrix}$$

where $h^+ = (h_1^+, h_2^+, \ldots h_{2m}^+)^T$ with $h_i^+ = \max(h_i, 0)$. Clearly, there are $n + 2m$ activation functions in the network. Correspondingly, there are $n + 2m$ artificial neurons in the network. The state of the neural network is $(x^T, y^T, z^T)^T \in \Re^{n+2m+r}$. The global convergence property of the neural network requires that $M > 0$.

As mentioned in Section I, if $M$ is symmetric, then LVI (1) is equivalent to the optimization problem (3), and many neural networks for convex quadratic optimization can be used to solve the LVI, e.g., the primal-dual neural network in [16] and an improved version in [19]. Because both neural networks can solve problems with equality constraints and bound constraints only, to deal with the inequality constraints $Ax \in \mathcal{Y}$ in (2), slack variables have to be used, which leads to $n + 2m + r$ state variables in both models. As the neural network in [19] is simpler than that in [16] in terms of the number of connections among neurons, we only present the former for solving (3) for comparison, i.e.,

$$\frac{d}{dt}\begin{pmatrix} x \\ z \end{pmatrix} = \lambda\begin{pmatrix} I & O \\ -B'' & I \end{pmatrix} \cdot \begin{pmatrix} -x + P_{\mathcal{X}\times\mathcal{Y}}\left(x - M''x - p'' + (B'')^T z\right) \\ -B''x + c'' \end{pmatrix} \quad (14)$$

where $\lambda > 0$, and

$$M'' = \begin{pmatrix} M & O \\ O & O \end{pmatrix} \quad p'' = \begin{pmatrix} p \\ O \end{pmatrix}$$

$$B'' = \begin{pmatrix} A & -I \\ B & O \end{pmatrix} \quad c'' = \begin{pmatrix} O \\ c \end{pmatrix}.$$

This neural network has $n + m$ activation functions (then $n + m$ neurons) and $n + 2m + r$ states.

In order to reduce the number of states of neural networks for quadratic programming, a *simplified dual neural network* was recently devised [22]. For solving (3), its state equation can be written as

$$\frac{du}{dt} = \lambda\Big\{ -A'''G(A''')^T u + P_{\mathcal{Y}\times\mathcal{X}} \times \left(A'''G(A''')^T u - u + A'''s\right) - A'''s\Big\} \quad (15)$$

where $\lambda > 0$, $A''' = (A^T, I)^T$, and

$$G = M^{-1} - M^{-1}B^T(BM^{-1}B^T)^{-1}BM^{-1}$$
$$s = M^{-1}\left(B^T(BM^{-1}B^T)^{-1}(BM^{-1}p + c) - p\right).$$

In neural network (15), the bound constraints are unified into inequality constraints $A'''x \in \mathcal{Y} \times \mathcal{X}$, which leads to fewer states in the model than in (14). Actually, (15) entails $n + m$ states only. However, both neural networks entail $n + m$ neurons. The disadvantage of (15) is that it requires $M$ to be positive definite, whereas (14) requires $M$ to be positive semidefinite only.

Neural network models (14) and (15) are believed to be two typical representatives for convex quadratic programming in terms of stability conditions and model complexity, respectively. It is possible to design a neural network with less than $n + m$ neurons or states for solving (1) based on differential inclusion theory [29]; nevertheless, the convergence of the network will definitely depend on some parameters, which are often inconvenient to choose.

It should be mentioned that the neural networks in [13], [14], [19], and [2] can solve LVI (1) via solving LVIs (7) and (8) under certain conditions. Following a similar analysis to what is done in this subsection, one can obtain the stability conditions and the number of neurons and states in the neural networks for solving these LVIs directly, which are summarized in Table I for comparison. In the table, the conditions in the first column are always stronger than the corresponding ones in the last column because of the presence of equality constraints in the LVI (1). Note that (7) is equivalent to (1) if $\bar{M} > 0$, and therefore, for solving (7), this condition is always required (see the second column in the table). Then, for a fair comparison for solving (1), one may focus on the last column because (8) is always

TABLE I
COMPARISON WITH SEVERAL SALIENT NEURAL NETWORKS IN THE LITERATURE FOR SOLVING LVIS (1), (7), AND (8) DIRECTLY

| | | LVI (1) | LVI (7) | LVI (8) |
|---|---|---|---|---|
| Refs. [13], [14] | Conditions | $M > 0$ | $\bar{M} > 0$ | $\bar{M} > 0$ |
| | Neurons | $n + 2m$ | $n + 2m$ | $n + 2m + r$ |
| | States | $n + 2m + r$ | $n + 2m + r$ | $n + 2m + r$ |
| Refs. [19] | Conditions | $M \geq 0, M^T = M$ | $\bar{M} > 0, M^T = M$ | $\bar{M} \geq 0, \bar{M}^T = \bar{M}$ |
| | Neurons | $n + m$ | $n + m$ | $n + m$ |
| | States | $n + 2m + r$ | $n + 2m + r$ | $n + 2m + r$ |
| Refs. [22] | Conditions | $M > 0, M^T = M$ | $\bar{M} > 0, M^T = M$ | $\bar{M} > 0, \bar{M}^T = \bar{M}$ |
| | Neurons | $n + m$ | $n + m$ | $n + m$ |
| | States | $n + m$ | $n + m$ | $n + m$ |
| Present paper | Conditions | – | – | $\bar{M} \geq 0$ |
| | Neurons | – | – | $n + m$ |
| | States | – | – | $n + m$ |

equivalent to (1). It can be concluded that, even if all neural networks are used to solve (8), the proposed neural network (12) is superior to others.

### D. Special Cases

In order to facilitate the design of GPNNs for users, we briefly present the results for some special cases. First, consider the LVI with $\Omega$ defined by

$$\Omega = \{x \in \Re^n | Ax \in \mathcal{Y}, x \in \mathcal{X}\} \tag{16}$$

where the parameters are same as in (2). By comparing the structure of this LVI with that of (8), we can design the following GPNN for solving the problem:

$$\frac{dw}{dt} = \lambda(\hat{M} + \hat{N})^T \left\{-\hat{N}w + P_{\mathcal{X}\times\mathcal{Y}}\left((\hat{N} - \hat{M})w - \hat{p}\right)\right\} \tag{17}$$

where $\lambda > 0$, and $w = (x^T, y^T)^T$ is the state variable that is of $n + m$ dimensions, and

$$\hat{M} = \begin{pmatrix} M & -A^T \\ O & I \end{pmatrix} \quad \hat{N} = \begin{pmatrix} I & O \\ A & O \end{pmatrix} \quad \hat{p} = \begin{pmatrix} p \\ O \end{pmatrix}.$$

The output of the neural network is $x(t)$, which is a part of the state variable $w(t)$. Clearly, for solving this LVI, neural networks (13) and (14) require more than $n + m$ states, and neural network (15) requires $n + m$ states exactly. The stability results of neural network (17) can be obtained from Theorem 3 by replacing $\bar{M}$ with $M$ in the statements.

Next, consider the LVI with $\Omega$ defined by

$$\Omega = \{x \in \Re^n | Bx = c, x \in \mathcal{X}\} \tag{18}$$

where the parameters are the same as in (2). It is easy to see that neural network (12) with all parameters unchanged, except for the following two:

$$\begin{aligned} \bar{A} &\rightarrow -B_I^{-1}B_{II} \\ \mathcal{V} &\rightarrow \left\{u \in \Re^r | \underline{x}_I - B_I^{-1}c \leq u \leq \overline{x}_I - B_I^{-1}c\right\} \end{aligned} \tag{19}$$

can be applied to solve the problem; and the convergence results can be stated as the same as that in Theorem 3. Moreover, the number of neurons (as well as states) of this neural network is $n$, which is equal to the dimension of the solution of the LVI.

If the equality constraints $Bx = c$ are present, and $M$ is already positive semidefinite, for designing a globally convergent neural network, it is not necessary to employ the substitution technique discussed in Section II to eliminate the equalities; instead, one can convert the equality constraints $Bx = c$ into inequality constraints $c \leq Bx \leq c$, and then formulate a neural network similar to (17). Of course, the resulting neural network will have more neurons than that formulated with the substitution technique. This could not be a serious problem, however, if there are only few equalities in the constraints, i.e., Rank$(B) = r \ll n$. Moreover, in the following, we show that the resulting neural network, in the case of $\Omega$ defined in (18), is a generalization of the primal-dual network in [16] for solving linear or quadratic programming problem (3) with this type of $\Omega$. Actually, from the above arguments, the state equation of the resulting neural network in this circumstance can be derived directly from (17) as

$$\begin{aligned} \frac{d}{dt}\begin{pmatrix} x \\ z \end{pmatrix} = \lambda \begin{pmatrix} M^T + I & B^T \\ -B & I \end{pmatrix} \\ \cdot \begin{pmatrix} -x + P_{\mathcal{X}}\left((I - M)x + B^T z - p\right) \\ -Bx + P_Z(Bx - z) \end{pmatrix} \end{aligned}$$

where $Z = \{z \in \Re^r | c \leq z \leq c\}$. By noticing that $P_Z$ is a constant $c$, the equation is rewritten as

$$\begin{aligned} \frac{d}{dt}\begin{pmatrix} x \\ z \end{pmatrix} = \lambda \begin{pmatrix} M^T + I & B^T \\ -B & I \end{pmatrix} \\ \cdot \begin{pmatrix} -x + P_{\mathcal{X}}\left((I - M)x + B^T z - p\right) \\ -Bx + c \end{pmatrix}. \end{aligned} \tag{20}$$

If $M$ is symmetric, this neural network is exactly the primal-dual network [16], although a special case of $\mathcal{X} = \{x \in \Re^n | x \geq 0\}$ is considered. An improved version of the primal-dual network can be found in [19] [see also (14)]. The GPNN (20) can be regarded as another extension of the primal-dual network as its global convergence requires the positive semidefiniteness of $M$ only while the symmetry condition is relaxed.

## IV. ILLUSTRATIVE EXAMPLES

We use two numerical examples to illustrate the effectiveness of the designed neural network in comparison with some other models.

*Example 1:* Consider an LVI (1) with $\Omega$ defined in (2), where

$$\begin{aligned} M &= \begin{pmatrix} -3 & -2 & -1 \\ -5 & 4 & 1 \\ 3 & 2 & 1 \end{pmatrix} \quad p = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} \\ \underline{x} &= (-5 \quad -5 \quad -5)^T \quad \overline{x} = (5 \quad 5 \quad 5)^T \\ A &= (1, -4, 4) \quad \underline{y} = -10 \quad \overline{y} = 10 \\ B &= (1, 1, -1)S \quad c = 5. \end{aligned}$$

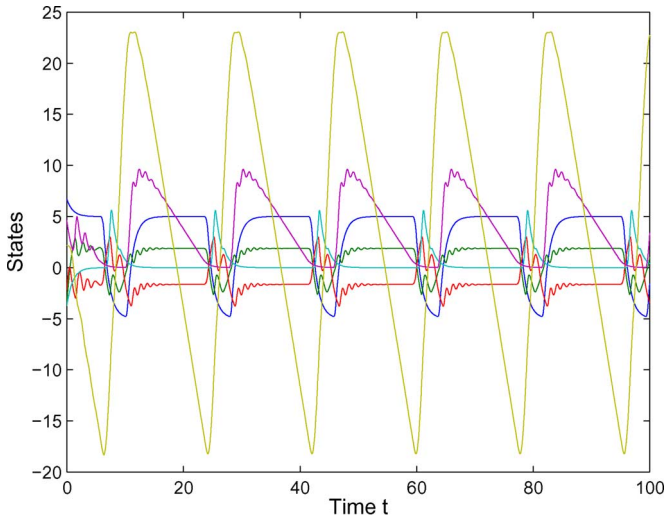


Fig. 1. Transient behavior of neural network (13) with a random initial point in Example 1.

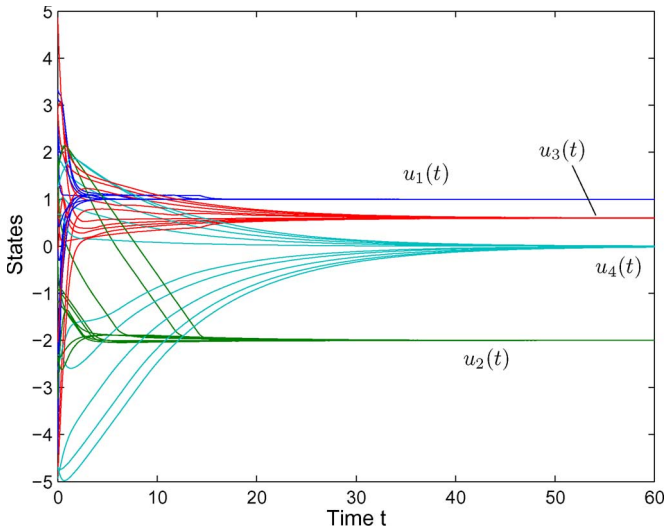


Fig. 2. Transient behavior of neural network (12) (state part) with ten random initial points in Example 1.

Since $M$ is asymmetric, singular, and indefinite, the neural networks (13)–(15) cannot be applied to solve the problem. Fig. 1 depicts the state trajectories of neural network (13) with $\lambda = 1$ from a random initial state. It is shown that the trajectories do not converge. Now we turn to the designed GPNN (12). Let $B_I = 1$ and $B_{II} = (1, -1)$, which follows

$$Q = \begin{pmatrix} -1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \bar{M} = Q^T M Q = \begin{pmatrix} 8 & 0 \\ 0 & 0 \end{pmatrix}.$$

According to Theorem 3, neural network (12) is globally convergent to a solution of the LVI. Numerical simulations verified this point. Fig. 2 illustrates the state trajectories of the neural network with $\lambda = 0.1$ from ten random initial states. It is shown that the trajectories converge to a steady state $(1.000, -2.000, 0.600, 0.000)^T$, which corresponds to the unique solution of the LVI $x^* = (2.000, 1.000, -2.000)^T$, by the output (12b).

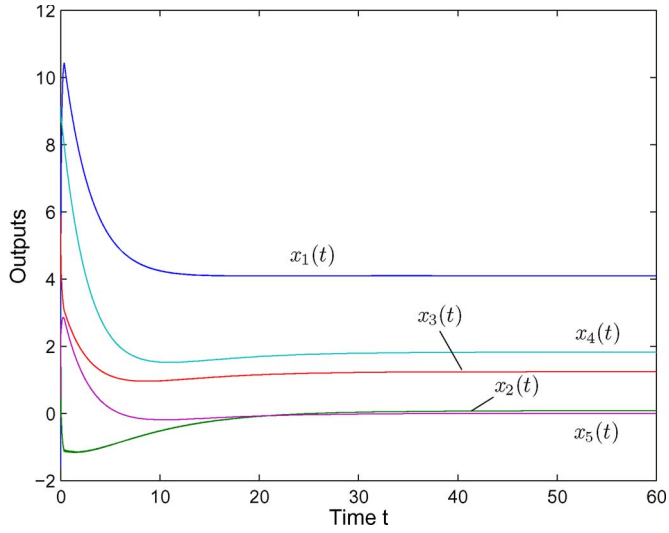


Fig. 3. Output trajectories of neural network (12b) with a random initial point in Example 2.

*Example 2:* Consider a quadratic optimization problem (3) with $\Omega$ defined in (18), where

$$M = \begin{pmatrix} 4 & 3 & -4 & -2 & 4 \\ 3 & 2 & 8 & -8 & 5 \\ -4 & 8 & 10 & 6 & -2 \\ -2 & -8 & 6 & 0 & -1 \\ 4 & 5 & -2 & -1 & -4 \end{pmatrix} \quad p = \begin{pmatrix} 3 \\ 0 \\ 2 \\ 6 \\ 0 \end{pmatrix}$$

$$\underline{x} = (0 \quad 0 \quad 0 \quad 0 \quad 0)^T, \quad \overline{x} = (10 \quad 10 \quad 10 \quad 10 \quad 10)^T$$

$$B = \begin{pmatrix} 1 & 0 & 3 & -1 & -2 \\ 0 & 1 & -3 & 2 & -2 \end{pmatrix} \quad c = \begin{pmatrix} 6 \\ 0 \end{pmatrix}.$$

Numerical simulation results show that none of the neural networks (13)–(15) globally converge to a steady state, which is due to the fact that $M$ is not positive semidefinite. Let

$$B_I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad B_{II} = \begin{pmatrix} 3 & -1 & -2 \\ -3 & 2 & -2 \end{pmatrix}.$$

Then

$$Q = \begin{pmatrix} -3 & 1 & 2 \\ 3 & -2 & 2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Q^T M Q = \begin{pmatrix} 82 & -29 & -3 \\ -29 & 28 & -33 \\ -3 & -33 & 80 \end{pmatrix} > 0.$$

This shows that $Mx + p$ is strictly (also strongly) monotone on $\Omega$ defined by (18), and as a result, there is only one solution $x^*$ to the problem, if it exists. According to Theorem 3, neural network (12) can be applied to solve the problem. All numerical simulations show that the neural network is globally convergent to $(4.096, 0.082, 1.242, 1.822, 0.000)^T$, i.e., the unique solution $x^*$. Fig. 3 depicts the output trajectories $x(t)$ of the neural network with $\lambda = 0.1$ from a random initial point.

*Example 3:* Consider the following $k$-winners-take-all (KWTA) problem:

$$x_i = f(\sigma_i) = \begin{cases} 1, & \text{if } \sigma_i \in \{k \text{ largest elements of } \sigma\} \\ 0, & \text{otherwise} \end{cases}$$

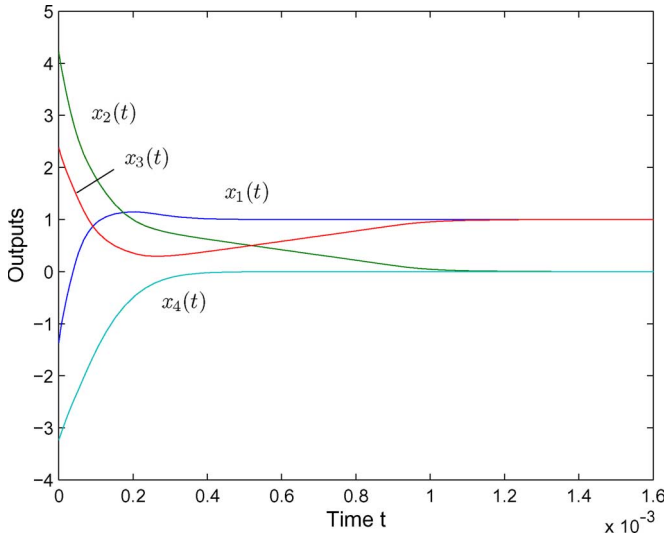


Fig. 4. Output trajectories of the general projection network for KWTA operation with a random initial point in Example 3.

where $\sigma \in \Re^n$ stands for the input vector, and $x \in \{0,1\}^n$ stands for the output vector. The KWTA operation accomplishes a task of selecting $k$ largest inputs from $n$ inputs in a network. A variety of applications as well as computational schemes of this problem can be found in [22] and references therein. In particular, in [22], the problem was formulated into a quadratic programming problem under the hypothesis that the $k$th largest input is not equal to the $(k+1)$th largest input, and a simplified dual neural network was developed for KWTA with $n$ neurons [cf. (15)]. Under the same hypothesis, similar to the arguments given in [22], the KWTA problem can be easily reasoned equivalent to the following linear programming problem:

$$\begin{array}{ll} \text{minimize} & -\sigma^T x \\ \text{subject to} & Bx = k, x \in [0,1]^n \end{array} \tag{21}$$

where $B = (1, \ldots, 1) \in \Re^{1\times n}$. Then, neural network (12) with the changes in (19) by setting $M = 0$, $p = -\sigma$, $c = k$, $\underline{x} = 0$, and $\overline{x} = 1$ can be used to solve the problem. Explicitly, the neural network can be written in the following form:

- State equation

$$\frac{d}{dt}\begin{pmatrix} u \\ v \end{pmatrix} = \lambda \begin{pmatrix} I & e^T \\ -e & I \end{pmatrix} \cdot \begin{pmatrix} -u + P_{\mathcal{U}}\left(u - e^T v + [-e^T, I]\sigma\right) \\ eu + P_{\mathcal{V}}(-eu - v) \end{pmatrix}$$

- Output equation

$$x = \begin{pmatrix} -e \\ I \end{pmatrix} u + \begin{pmatrix} k \\ O \end{pmatrix}$$

where $e = (1, \ldots, 1) \in \Re^{1\times(n-1)}$, $\mathcal{U} = \{u \in \Re^{n-1} | 0 \le u \le 1\}$, and $\mathcal{V} = \{v \in \Re | -k \le v \le 1-k\}$. Clearly, this network also entails $n$ neurons. However, in contrast to [22], it is seen that no parameter is needed to choose in this network.

In the above KWTA problem, let $k = 2$, $n = 4$, and the inputs $\sigma_1 = 5.9$, $\sigma_2 = 4.0$, $\sigma_3 = 4.2$, and $\sigma_4 = -3$. Fig. 4 shows the output trajectories of the GPNN with $\lambda = 10^4$ from a random initial point. It is seen that the trajectories converge to a point $(1, 0, 1, 0)^T$, which implies $\sigma_1$ and $\sigma_3$ should be selected.

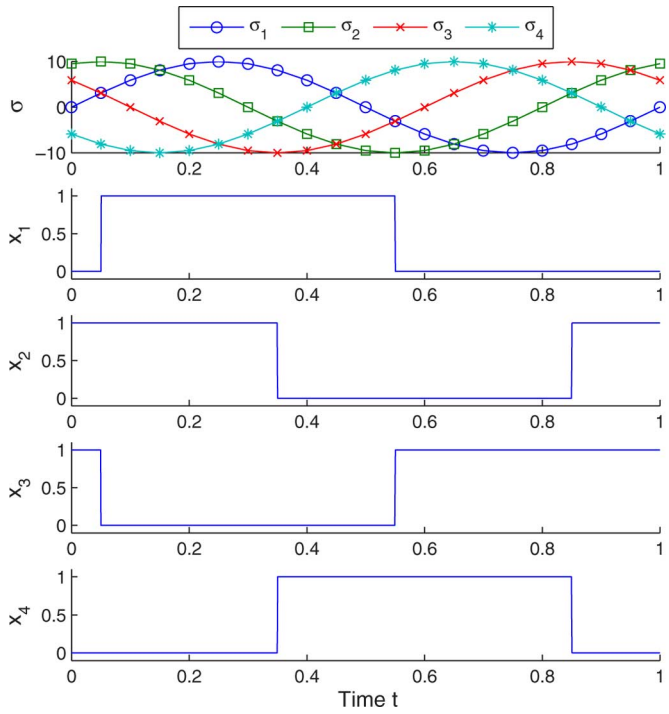


Fig. 5. Inputs and outputs of the GPNN in Example 3.

Next, let $k = 2$, $n = 4$, and the inputs be time-varying signals $v_i = 10\sin[2\pi(t + 0.2(I-1))], t \in [0,1]\ \forall i = 1,2,3,4$ (see the top subfigure in Fig. 5). The other four subfigures in Fig. 5 record the outputs of the network at each time instant $t$. It is readily checked that the outputs are correct.

## V. Concluding Remark

In this correspondence, we have presented a design method of the GPNN for solving a class of LVIs with linear equality and two-sided linear inequality constraints. The neural network is globally convergent to a solution of the problem under the condition that the linear mapping $Mx + p$ is monotone on the constrained set $\Omega$. As this condition is weaker than $M$ being positive semidefinite, which is always required by existing neural networks in the literature, the designed neural network is superior in terms of stability conditions. Moreover, the designed model is of lower structural complexity than most existing ones. Special cases are discussed for solving LVIs with different types of linear constraints. Several numerical examples are discussed to illustrate the good performance of the designed neural networks.

## References

[1] P. T. Harker and J. S. Pang, "Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications," *Math. Program.*, vol. 48, no. 1–3, pp. 161–220, Mar. 1990.

[2] F. Facchinei and J. S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*, vol. I and II. New York: Springer-Verlag, 2003.

[3] B. He, "A new method for a class of linear variational inequalities," *Math. Program.*, vol. 66, no. 1–3, pp. 137–144, Aug. 1994.

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

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

[6] B. He and H. Yang, "A neural-network model for monotone linear asymmetric variational inequalities," *IEEE Trans. Neural Netw.*, vol. 11, no. 1, pp. 3–16, Jan. 2000.

[7] X. Liang and J. Si, "Global exponential stability of neural networks with globally Lipschitz continuous activations and its application to linear variational inequality problem," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 349–359, Mar. 2001.

[8] Y. Xia and J. Wang, "On the stability of globally projected dynamical systems," *J. Optim. Theory Appl.*, vol. 106, no. 1, pp. 129–150, Jul. 2000.

[9] Y. Xia, H. Leung, and J. Wang, "A projection neural network and its application to constrained optimization problems," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 49, no. 4, pp. 447–458, Apr. 2002.

[10] Y. Xia, "Further results on global convergence and stability of globally projected dynamical systems," *J. Optim. Theory Appl.*, vol. 122, no. 3, pp. 627–649, Sep. 2004.

[11] X. Hu and J. Wang, "Solving pseudomonotone variational inequalities and pseudoconvex optimization problems using the projection neural network," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1487–1499, Nov. 2006.

[12] X. Hu and J. Wang, "A recurrent neural network for solving a class of general variational inequalities," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 3, pp. 528–539, Jun. 2007.

[13] Y. Xia, "An extended projection neural network for constrained optimization," *Neural Comput.*, vol. 16, no. 4, pp. 863–883, Apr. 2004.

[14] Y. Xia, "On convergence conditions of an extended projection neural network," *Neural Comput.*, vol. 17, no. 3, pp. 515–525, Mar. 2005.

[15] X. B. Gao, L. Z. Liao, and L. Q. Qi, "A novel neural network for variational inequalities with linear and nonlinear constraints," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1305–1317, Nov. 2005.

[16] Y. Xia, "A new neural network for solving linear and quadratic programming problems," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1544–1547, Nov. 1996.

[17] Y. Leung, K.-Z. Chen, Y.-C. Jiao, X.-B. Gao, and K. S. Leung, "A new gradient-based neural network for solving linear and quadratic programming problems," *IEEE Trans. Neural Netw.*, vol. 12, no. 5, pp. 1074–1083, Sep. 2001.

[18] 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, Nov. 2001.

[19] Q. Tao, J. Cao, M. Xue, and H. Qiao, "A high performance neural network for solving nonlinear programming problems with hybrid constraints," *Phys. Lett. A*, vol. 288, no. 2, pp. 88–94, 2001.

[20] Y. 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, Sep. 2004.

[21] Q. Liu, J. Cao, and Y. Xia, "A delayed neural network for solving linear projection equations and its analysis," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 834–843, Jul. 2005.

[22] 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.

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

[24] Y. 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.

[25] X. Hu and J. Wang, "Solving generally constrained generalized linear variational inequalities using the general projection neural networks," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, Nov. 2007.

[26] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. New York: Academic, 1982.

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

[28] J. S. Pang and J. C. Yao, "On a generalization of a normal map and equations," *SIAM J. Control Optim.*, vol. 33, no. 1, pp. 168–184, Jan. 1995.

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