A Novel Neural Network for a Class of Convex Quadratic Minimax Problems

Gao, Xing Bao; Liao, Li Zhi

Published in:
Neural Computation

DOI:
10.1162/neco.2006.18.8.1818

Published: 01/08/2006

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Gao, X. B., & Liao, L. Z. (2006). A Novel Neural Network for a Class of Convex Quadratic Minimax Problems. *Neural Computation*, *18*(8), 1818-1846. https://doi.org/10.1162/neco.2006.18.8.1818

A Novel Neural Network for a Class of Convex Quadratic Minimax Problems

Xing-Bao Gao
xinbaog@snnu.edu.cn
College of Mathematics and Information Science, Shaanxi Normal University, Xi'an, Shaanxi 710062, China

Li-Zhi Liao
liliao@hkbu.edu.hk
Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong, China

Based on the inherent properties of convex quadratic minimax problems, this article presents a new neural network model for a class of convex quadratic minimax problems. We show that the new model is stable in the sense of Lyapunov and will converge to an exact saddle point in finite time by defining a proper convex energy function. Furthermore, global exponential stability of the new model is shown under mild conditions. Compared with the existing neural networks for the convex quadratic minimax problem, the proposed neural network has finite-time convergence, a simpler structure, and lower complexity. Thus, the proposed neural network is more suitable for parallel implementation by using simple hardware units. The validity and transient behavior of the proposed neural network are illustrated by some simulation results.

1 Introduction

In this letter, we are interested in the following convex quadratic minimax problem:

$$\min_{x\in U}\{\max_{y\in V}\{f(x,y)\}\}, \tag{1.1}$$

where

$$f(x,y)=\frac{1}{2}x^T Hx+h^T x-x^T Qy-\frac{1}{2}y^T Sy-s^T y, \tag{1.2}$$

$H\in R^{m\times m}$, $S\in R^{n\times n}$, $Q\in R^{m\times n}$, $h\in R^m$, and $s\in R^n$ are given with H and S being symmetric and positive semidefinite, $U=\{x\in R^m \mid a_i\le x_i\le b_i,$

$i = 1, 2, \cdots, m\}$, $V = \{y \in R^n \mid c_j \leq y_j \leq d_j,\ \ j = 1, 2, \cdots, n\}$, and some $-a_i$ (or $-c_j, b_i, d_j$) could be $+\infty$.

Minimax problems provide a useful reformulation of optimality conditions and also arise in a variety of engineering and economic contexts, including game theory, military scheduling, and automatic control. In particular, problem 1.1 includes:

- (Piecewise) linear programming ($H = 0$ and $S = 0$)
- Linear programming ($H = 0$, $S = 0$, and $Q = 0$)
- Quadratic programming ($S = 0$ and $H \neq 0$)
- Linear and quadratic programming with bound constraints

In many engineering and scientific applications, real-time online solutions of minimax problems are desired. However, traditional algorithms (Fukushima, 1992; He, 1996, 1999; Rockafellar, 1987; Solodov & Tseng, 1996; Tseng, 2000) are not suitable for a real-time online implementation on the computer because 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 problems with high dimension and dense structure is to employ artificial neural network–based circuit implementation. Because of the dynamic nature of 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 done in parallel. Therefore, the neural network approach can solve optimization problems in running times that are orders of magnitude much faster than conventional optimization algorithms executed on general-purpose digital computers. It is of great interest to develop some neural network models that could provide a real-time online solution.

In recent years, the neural network approach for solving optimization problems has been studied by many researchers, and many good results have been achieved (Bouzerdorm & Pattison, 1993; Friesz, Bernstein, Mehta, Tobin, & Ganjlizadeh, 1994; Gao, 2003, 2004; Gao & Liao, 2003; Gao, Liao, & Xue, 2004; Han, Liao, Qi, & Qi, 2001; He & Yang, 2000; Xia, 2004; Xia & Feng, 2004; Xia, Feng, & Wang, 2004; Xia & Wang, 1998, 2000, 2001). Since the condition of the saddle point of equation 1.2 can be formulated as the following linear variational inequality LVI(M, q, C), to find a vector $z^* \in C$ such that

$$(z - z^*)^T (Mz^* + q) \geq 0, \quad \forall z \in C, \tag{1.3}$$

where $M \in R^{k\times k}$, $q \in R^k$, and $C \subseteq R^k$ is a nonempty closed convex set (see remark 1), problem 1.1 can be solved by using the models in Gao et al., (2004), He and Yang (2000), and Xia and Wang (1998, 2000). In particular, Gao et al. (2004) proposed the following neural network,

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\lambda \begin{pmatrix} x - P_U\big(x - Hx - h + Qy\big) \\ y - P_V\big(y - Sy - s - Q^T x\big) \end{pmatrix}, \tag{1.4}$$

where $\lambda > 0$ is a scaling constant, $P_U : R^m \to U$ is the projection operator defined by

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

$\|\cdot\|$ is the Euclidean norm, and $P_V : R^n \to V$ is the projection operator defined similar to P_U. Gao et al. (2004) also provided several simple and feasible sufficient conditions to ensure the asymptotical stability of equation 1.4. Although model 1.4 has a one-layer structure and is exponentially stable for any initial point in $U \times V$ when matrices H and S are positive definite, its convergence is not very satisfactory since it may not be stable and does not have a finite-time convergence when matrices H and S are only positive semidefinite. For example, for the following problem,

$$\min_x \max_y (xy), \tag{1.5}$$

where x and $y \in R^1$, model 1.4 can be simplified as

$$\begin{cases} \dfrac{dx}{dt} = \lambda y, \\ \dfrac{dy}{dt} = -\lambda x. \end{cases} \tag{1.6}$$

It is easy to see that equation 1.6 is divergent. The models proposed by He and Yang (2000) and Xia and Wang (1998) have good stability performance; however, because the model in He and Yang (2000) is not suitable for parallel implementation due to the choice of the varying parameter and the model in Xia and Wang (1998) has a complex structure, further simplification can be achieved. Therefore, it is necessary to build a new neural network for equation 1.1 with a lower complexity and good stability and convergence results.

Based on the above considerations, in this article, we will propose a new neural network model for solving problem 1.1 by means of sufficient and necessary conditions of the saddle point of equation 1.2, define a convex energy function by introducing a convex function, and prove that the proposed neural network is stable in the sense of Lyapunov and will converge to an exact saddle point in finite time when matrices H and S are only positive semidefinite. Furthermore, global exponential stability of the new model is also shown when matrices H and S are positive definite. Compared with the existing neural networks and some conventional numerical methods, the new model has a lower complexity and finite-time convergence, and its asymptotical stability requires only the positive semidefiniteness of matrices H and S. Thus, the new model is very simple and more suitable for the hardware implementation.

The solution of problem 1.1 is closely related to the saddle point of $f(x, y)$. A point $(x^*, y^*) \in U \times V$ is said to be a saddle point of $f(x, y)$ if

$$f(x^*, y) \leq f(x^*, y^*) \leq f(x, y^*), \quad \forall (x, y) \in U \times V. \tag{1.7}$$

Throughout this letter, we assume that the set $K^* = \{(x, y) \in U \times V \mid (x, y)$ is a saddle point of $f(x, y)\} \neq \emptyset$ and there exists a finite point $(x^*, y^*) \in K^*$. Obviously, if $(x^*, y^*) \in K^*$ is a saddle point of $f(x, y)$, then it must be a solution of problem 1.1. Therefore, it would be sufficient to find a saddle point of $f(x, y)$ for problem 1.1.

For the convenience of later discussions, it is necessary to introduce the following definition:

Definition 1. *A neural network is said to have finite-time convergence to one of its equilibrium points z^* if there exists a time τ_0 such that the output trajectory $z(t)$ of this network reaches z^* for $t \geq \tau_0$ (see Xia et al., 2004).*

In our following discussions, we let $\|\cdot\|$ denote the Euclidean norm, I_n denote the identity matrix of order n, $\nabla\varphi(x) = (\partial\varphi(x)/\partial x_1, \partial\varphi(x)/\partial x_2, \ldots, \partial\varphi(x)/\partial x_n)^T \in R^n$ denote the gradient vector of the differentiable function $\varphi(x)$ at x. For any vector $u \in R^n$, u^T denotes its transpose. For any $n \times n$ real symmetric matrix M, $\lambda_{\min}(M)$ and $\lambda_{\max}(M)$ denote its minimum and maximum eigenvalues, respectively. A basic property of the projection mapping on a closed convex set $U \subseteq R^m$ is (Kinderlehrer & Stampacchia, 1980)

$$[w - P_U(w)]^T [P_U(w) - p] \geq 0, \quad \forall w \in R^m, \ p \in U. \tag{1.8}$$

The rest of the letter is organized as follows. In section 2, a neural network model for problem 1.1 is proposed. The stability and convergence of the proposed network are analyzed in section 3. The simulation results of our proposed neural network are reported in section 4. Finally, some concluding remarks are drawn in section 5.

2 A Neural Network Model

In this section, a neural network for solving problem 1.1 is presented, and the comparisons with the existing neural networks and some conventional numerical methods are discussed. First, we provide a necessary and sufficient condition for the saddle point of $f(x, y)$ in equation 1.2. This result provides the theoretical foundation for us to design the neural network for problem 1.1.

Theorem 1. $(x^*, y^*) \in K^*$ *if and only if*

$$\begin{cases} (x - x^*)^T(Hx^* + h - Qy^*) \geq 0, & \forall x \in U, \\ (y - y^*)^T(Sy^* + s + Q^T x^*) \geq 0, & \forall y \in V. \end{cases} \tag{2.1}$$

Proof. From equation 1.7 and theorem 3.3.3 in Bazaraa, Sherali, and Shetty (1993), this can be easily proved.

Remark 1. Theorem 1 indicates that $z^* = ((x^*)^T, (y^*)^T)^T \in K^*$ if and only if it is a solution of a monotone LVI(M, q, C) defined in equation 1.3 with $k = m + n$,

$$M = \begin{pmatrix} H & -Q \\ Q^T & S \end{pmatrix}, \quad q = \begin{pmatrix} h \\ s \end{pmatrix}, \quad \text{and} \quad C = U \times V. \tag{2.2}$$

From equations 1.8 and 2.1, we can easily establish the following result, which shows that a saddle point (x^*, y^*) of $f(x, y)$ in equation 1.2 is the projection of some vector on $U \times V$.

Lemma 1. $(x^*, y^*) \in K^*$ *if and only if*

$$\begin{cases} x^* = P_U(x^* - Hx^* - h + Qy^*), \\ y^* = P_V(y^* - Sy^* - s - Q^T x^*), \end{cases} \tag{2.3}$$

where $P_U(x) = [(P_U(x))_1, (P_U(x))_2, \ldots, (P_U(x))_m]^T$ *and* $(P_U(x))_i = min\{b_i, max\{x_i, a_i\}\}$ *for* $i = 1, 2, \ldots, m$, $P_V(y) = [(P_V(y))_1, (P_V(y))_2, t, (P_V(y))_n]^T$, *and* $(P_V(y))_j = min\{d_j, max\{y_j, c_j\}\}$ *for* $j = 1, 2, \ldots, n$.

Lemma 1 indicates that a saddle point (x^*, y^*) of $f(x, y)$ in equation 1.2 can be obtained by solving equation 2.3. Based on the above results, we propose the following dynamical system for a neural network model to solve problem 1.1:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\lambda \begin{pmatrix} 2\big(x - P_U\big(x - Hx - h + QP_V\big(y - Sy - s - Q^T x\big)\big)\big) \\ y - P_V\big(y - Sy - s - Q^T x\big) \end{pmatrix}, \tag{2.4}$$

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

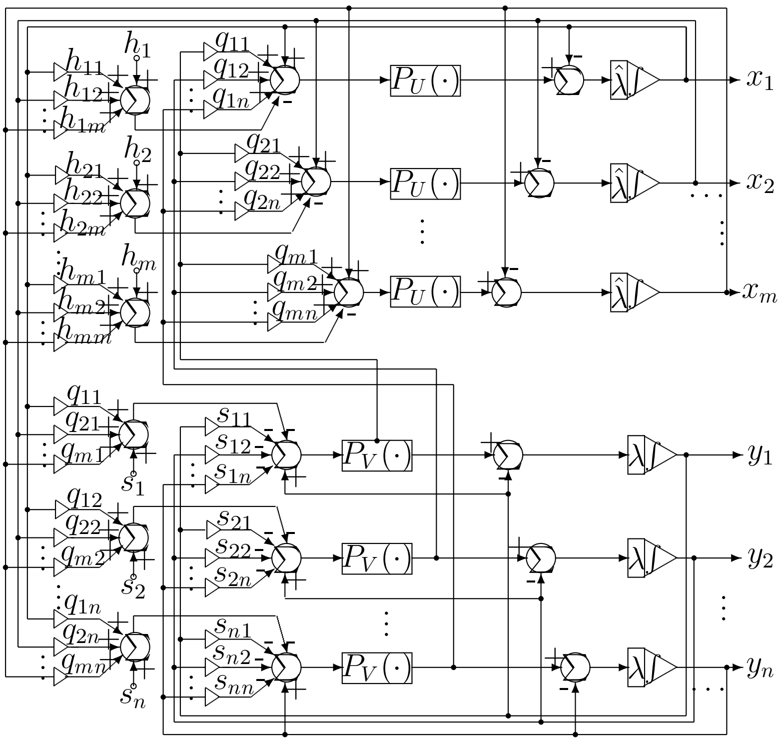


Figure 1: The architecture of network 2.4.

As a result of lemma 1, we have the following result, which describes the relationship between an equilibrium point of equation 2.4 and a saddle point of $f(x, y)$ in equation 1.2.

Lemma 2. $(x^T, y^T)^T \in K^*$ *if and only if* (x, y) *is an equilibrium point of network 2.4.*

Proof. From lemma 1 and equations 2.3 and 2.4, the result is trivial.

Lemma 2 also illustrates that a saddle point (x^*, y^*) of $f(x, y)$ in equation 1.2 is the projection of some vector on $U \times V$ and can be obtained by the equilibrium point of equation 2.4.

The architecture of neural network 2.4 is shown in Figure 1, where vectors x and y are the network's outputs, vectors $h = (h_1, h_2, \ldots, h_m)^T$ and $s = (s_1, s_2, \ldots, s_n)^T$ are the external inputs, the projection operators $P_U(\cdot)$ and $P_V(\cdot)$ could be implemented by some piecewise activation functions

(Bouzerdorm & Pattison, 1993), and other parameters are defined by $H = (h_{ij})_{m\times m}$, $Q = (q_{ij})_{m\times n}$, $S = (s_{ij})_{n\times n}$, and $\hat{\lambda} = 2\lambda$. According to Figure 1, the circuit realizing the proposed neural network 2.4 consists of $m+n$ integrators, $m+n$ linear piecewise activation functions, $(m+n)^2$ weighted connections for H, S, Q, and Q^T, and $(m+n)^2 + 2(m+n)$ adders. Thus, it can be implemented by using simple hardware units.

For the convenience of later discussions, we denote $z = (x^T, y^T)^T \in R^{m+n}$ and

$$\begin{cases} v = P_V\big(y - Sy - s - Q^T x\big), \\ u = P_U(x - Hx - h + Qv). \end{cases} \tag{2.5}$$

It should be noted that the definition of u in equation 2.5 requires the value of v. Then the proposed neural network 2.4 can be written as

$$\frac{dz}{dt} = -\lambda F(z) = -\lambda \begin{pmatrix} 2(x-u) \\ y - v \end{pmatrix}. \tag{2.6}$$

To show the advantages of the proposed neural network 2.4, we compare it with four existing neural network models and some conventional numerical methods. First, we look at model 1.4 proposed by Gao et al. (2004). The function $F(z)$ in equation 2.6 and the right-hand side of equation 1.4 are totally different since $u = P_U(x - Hx - h + Qv) \neq P_U(x - Hx - h + Qy)$. It is easy to see that the complexity of the above model is about the same as that of proposed neural network 2.4, yet the stability conditions are different. When matrices H and S are only positive semidefinite, theorem 3 ensures that neural network 2.4 is stable in the sense of Lyapunov and converges to a saddle point in finite time, but model 1.4 may not be stable and may not converge even when the initial point z^0 lies in $U \times V$ (see examples 2–5 in section 4). Thus the stability and finite-time convergence conditions of model 1.4 are stronger than that of 2.4. To clarify this issue further, we consider problem 1.5; then model 2.4 can be written as

$$\begin{cases} \frac{dx}{dt} = 2\lambda(y - x), \\ \frac{dy}{dt} = -\lambda x. \end{cases}$$

Obviously this system differs from model 1.6, and is stable and convergent, but system 1.6 is divergent.

Second, we compare the proposed neural network 2.4 with the models proposed by He and Yang (2000) and Xia and Wang (1998). The model proposed by Xia and Wang (1998) for problem 1.1 is defined as

$$\frac{dz}{dt} = -\lambda\big(I_{m+n} + M^T\big)e(z) \tag{2.7}$$

where $\lambda > 0$ is a scaling constant, M and q are defined in equation 2.2, and

$$e(z) = z - P_{U\times V}(z - Mz - q). \tag{2.8}$$

In terms of the model complexity, it is easy to see that the total multiplications/divisions and additions/subtractions per iteration for equation 2.7 are $2(m+n)^2 + m + n$ and $2(m+n)^2 + 2(m+n)$, respectively. But the total multiplications/divisions and additions/subtractions per iteration for neural network 2.4 are $(m+n)^2 + 2m + n$ and $(m+n)^2 + 2(m+n)$, respectively. Thus the asymptotic complexity of model 2.4 is about half of model 2.7. Furthermore, for problem 1.1, the model proposed by He and Yang (2000) is

$$\frac{dz}{dt} = \lambda\{P_{U\times V}[z - \theta\alpha(z)(M^T e(z) + Mz + q)] - z\}, \tag{2.9}$$

where $\lambda > 0$ is a scaling constant, $\theta \in (0, 2)$, M, q, and $e(z)$ are defined in equations 2.2 and 2.8, respectively, and $\alpha(z) = \|e(z)\|^2/\|(I_{m+n} + M^T)e(z)\|^2$. It is easy to see that this model is not suitable for parallel implementation due to the choice of the varying parameter $\alpha(z)$ and requires computing two projections and terms $e(z)$, $M^T e(z)$, and $\alpha(z)$ per iteration. Even though the proposed neural network 2.4 has a two-layer structure, it is required to compute only one projection and term $F(z)$ in equation 2.6 per iteration. Since the complexity of $F(z)$ is about the same as that of $e(z)$, the proposed neural network has a low complexity. Therefore, model 2.4 is simpler than models 2.7 and 2.9 and reduces the model complexity in implementation.

In addition, no result for the finite-time convergence for models 2.7 and 2.9 is available in the literature, and the stability of model 2.9 requires that the initial point z^0 lies in $U \times V$, yet theorem 3 ensures that neural network 2.4 is stable and convergent in finite time for any initial point $z^0 \in R^{m+n}$.

Third, we compare the proposed neural network 2.4 with the model proposed by Gao (2004). Model 2.4 is designed to solve convex quadratic minimax problems, while the model in Gao (2004) is developed to solve nonlinear convex programming problems. Thus, the energy functions and theoretical results of the two models are different. In particular, model 2.4 is globally exponentially stable when matrices H and S are positive definite (see theorem 4), but the model in Gao (2004) has no exponential stability result. Moreover, for a convex quadratic problem (problem 1.1) with $S = 0$ and $V = \{y \in R^m | y \geq 0\}$), even though the two models are the same, the finite-time convergence results are different. Model 2.4 is stable and converges to a saddle point in finite time (see theorem 3), but no result for the finite-time convergence of the model in Gao (2004) is available in the literature.

Finally, we compare the proposed neural network 2.4 with two typically numerical methods: a modified projection-type method (Solodov & Tseng, 1996) and a forward-backward splitting method (Tseng, 2000). For problem 1.1, a modified projection-type method proposed by Solodov and Tseng (1996) is defined as

$$z^{k+1} = z^k - \theta\gamma(z^k)N^{-1}(I_{m+n} + M^T)e(z^k), \tag{2.10}$$

where $\theta \in (0, 2)$, N is an $(m+n) \times (m+n)$ symmetric positive-definite matrix, M and $e(z)$ are defined in equations 2.2 and 2.8, respectively, and $\gamma(z) = \|e(z)\|^2/[e(z)^T(I_{m+n} + M)N^{-1}(I_{m+n} + M^T)e(z)]$. It is easy to see that this method is not suitable for parallel implementation due to the choice of the varying parameter $\gamma(z^k)$, and its asymptotic complexity is about two times that of model 2.4 even when $N = I_{m+n}$. Furthermore, for problem 1.1, a forward-backward splitting method proposed by Tseng (2000) is defined as

$$\begin{cases} \bar{z}^k = P_{U\times V}\left[z^k - \theta\left(Mz^k + q\right)\right], \\ z^{k+1} = P_{U\times V}\left[\bar{z}^k + \theta M\left(z^k - \bar{z}^k\right)\right], \end{cases} \tag{2.11}$$

where θ is a positive constant and M and q are defined in equation 2.2. This method can be viewed as a prediction-correction method and requires two projections per iteration, and its asymptotic complexity is about two times that of model 2.4. In addition, besides the positive semidefiniteness requirement for matrix M, the parameters N and θ are key to the convergence of method 2.10, and method 2.11 is globally convergent only when $\theta < \nu/\|M\|$ with $0 < \nu < 1$. On the other hand, the stability and convergence of model 2.4 require only the positive semidefiniteness of matrices H and S, and model 2.4 has finite-time convergence without condition $\theta < \nu/\|M\|$ with $0 < \nu < 1$. Thus, model 2.4 is simpler than methods 2.10 and 2.11, avoids the difficulty of choosing the network parameters, and requires weaker convergence condition.

3 Stability Analysis

In this section, we study some theoretical properties for model 2.4. First, we prove the following lemma, which will be very useful in our later discussion.

Lemma 3. *Let*

$$\varphi(z) = \frac{1}{2}(\|y - Sy - s - Q^T x\|^2 - \|y - Sy - s - Q^T x - v\|^2), \tag{3.1}$$

where v is defined in equation 2.5. Then the following is true:

i. *$\varphi(z)$ is continuously differentiable and convex on R^{m+n}.*

ii. *For any $z, z' = ((x')^T, (y')^T)^T \in R^{m+n}$, the following inequality holds:*

$$\varphi(z) \le \varphi(z') + (z - z')^T \nabla\varphi(z') + (z - z')^T W(z - z')/2,$$

where

$$W = \begin{pmatrix} QQ^T & -Q(I_n - S) \\ -(I_n - S)Q^T & (I_n - S)^2 \end{pmatrix} = \begin{pmatrix} -Q \\ I_n - S \end{pmatrix} \begin{pmatrix} -Q^T & I_n - S \end{pmatrix}. \tag{3.2}$$

Proof. From equation 1.8, we can easily verify that for any closed convex set Ω,

$$\|P_\Omega(p) - P_\Omega(w)\|^2 \le (p - w)^T [P_\Omega(p) - P_\Omega(w)] \le \|p - w\|^2, \quad \forall p, w \in R^n. \tag{3.3}$$

Let

$$\varphi_1(z) = \|y - Sy - s - Q^T x\|^2/2$$

and

$$\varphi_2(z) = \|y - Sy - s - Q^T x - v\|^2/2,$$

where v is defined in equation 2.5. Then $\varphi(z) = \varphi_1(z) - \varphi_2(z)$.

i. Obviously $\varphi_2(z)$ is a compound function of the two functions: $\psi(w) = \|w - P_V(w)\|^2/2$ and $w = y - Sy - s - Q^T x$. According to lemma 3.7 in Smith, Friesz, Bernstein, and Suo (1997), the function $\psi(w)$ is continuously differentiable and $\nabla\psi(w) = w - P_V(w)$. Thus, their compound function $\varphi_2(z)$ is differential with respect to z, and

$$\nabla\varphi_2(z) = \begin{pmatrix} -Q\big(y - Sy - s - Q^T x - v\big) \\ (I_n - S)\big(y - Sy - s - Q^T x - v\big) \end{pmatrix}. \tag{3.4}$$

Therefore, $\varphi(z)$ defined in equation 3.1 is also continuously differentiable and

$$\nabla\varphi(z) = \begin{pmatrix} -Qv \\ (I_n - S)v \end{pmatrix}. \tag{3.5}$$

For any z, $z' \in R^{m+n}$, let $v' = P_V(y' - Sy' - s - Q^T x')$. Then from equation 3.5, we have

$$
\begin{aligned}
(z - z')^T[\nabla\varphi(z) - \nabla\varphi(z')] = (v - v')^T[(I_n - S)(y - y')\\
-Q^T(x - x')] \geq \|v - v'\|^2,
\end{aligned}
$$

where the last step follows by setting $p = y - Sy - s - Q^T x$ and $w = y' - Sy' - s - Q^T x'$ on the left-hand side of equation 3.3. Thus, $\varphi(z)$ is convex on R^{m+n} by theorem 3.4.5 in Ortega and Rheinboldt (1970).

ii. Similar to the proof of lemma 3i, we can prove that $\varphi_2(z)$ is also convex on R^{m+n} by equation 3.4 and the right-hand side of equation 3.3. Thus, $\forall z$, $z' \in R^{m+n}$; we have

$$
\varphi_1(z) = \varphi_1(z') + (z - z')^T\nabla\varphi_1(z') + \frac{1}{2}(z - z')^T W(z - z'),
$$

and

$$
\varphi_2(z) \geq \varphi_2(z') + (z - z')^T\nabla\varphi_2(z')
$$

from theorem 3.3.3 in Bazaraa et al. (1993). Therefore, lemma 3ii holds from $\varphi(z) = \varphi_1(z) - \varphi_2(z)$.

Remark 2. If V is a closed convex cone, for example, $V = \{y \in R^n | y \geq 0\}$, then $\varphi(z) = \|v\|^2/2$ (v is defined in equation 2.5) is continuously differentiable on R^n. However, this may not be true for a general closed convex set V. For example, let $V = \{x \in R^1 | -1 \leq x \leq 1\}$; then

$$
v^2 = [P_V(x)]^2 = \begin{cases} 1, & \text{if} \quad x > 1, \\ x^2, & \text{if} \quad -1 \leq x \leq 1, \\ 1, & \text{if} \quad x < -1, \end{cases}
$$

and

$$
2\varphi(x) = x^2 - [x - P_V(x)]^2 = \begin{cases} 2x - 1, & \text{if} \quad x > 1, \\ x^2, & \text{if} \quad -1 \leq x \leq 1, \\ -2x - 1, & \text{if} \quad x < -1. \end{cases}
$$

Thus $[P_V(x)]^2 \neq x^2 - [x - P_V(x)]^2$, and $[P_V(x)]^2$ is not differentiable on $(-\infty, +\infty)$.

From lemma 3i, we can define the function

$$
\begin{aligned}
G(z, z^*) = \frac{1}{2}[(x - x^*)^T H(x - x^*) + 3(y - y^*)^T S(y - y^*)]\\
+ \frac{1}{2}\|z - z^*\|^2 + \varphi(z) - \varphi(z^*) - (z - z^*)^T\nabla\varphi(z^*), \qquad (3.6)
\end{aligned}
$$

where $z^* \in K^*$ is finite and $\varphi(z)$ is defined in equation 3.1. Then we have the following result, which explores some properties for $G(z, z^*)$.

Lemma 4. *Let $G(z, z^*)$ be the function in equation 3.6 and W be the matrix in equation 3.2. Then the following is true:*

i. *$G(z, z^*)$ is continuously differentiable and convex on R^{m+n} and*

$$\frac{1}{2}\|z - z^*\|^2 \leq G(z, z^*) \leq \frac{\mu_1}{2}\|z - z^*\|^2, \quad \forall z \in R^{m+n}, \tag{3.7}$$

where $\mu_1 = 1 + \lambda_{max}(W) + max\{\lambda_{max}(H), 3\lambda_{max}(S)\} \geq 1$.

ii. $\nabla G(z, z^*)^T F(z) \geq \|F(z)\|^2/2, \forall z \in R^{m+n}$.

iii. $\nabla G(z, z^*)^T F(z) \geq 2\mu_2 G(z, z^*), \quad \forall z \in R^{m+n},$ *where* $\mu_2 = 2min\{\lambda_{min}(H), \lambda_{min}(S)\}/\mu_1$.

Proof. i. Obviously, $\mu_1 \geq 1$ by the semidefiniteness of matrices W, H, and S. From lemma 3i, we know that $\varphi(z)$ is continuously differentiable and convex. Thus, $G(z, z^*)$, defined in equation 3.6, is also continuously differentiable and convex on R^{m+n}, and

$$\varphi(z) \geq \varphi(z^*) + (z - z^*)^T \nabla\varphi(z^*), \qquad \forall z \in R^{m+n}$$

from theorem 3.3.3 in Bazaraa et al. (1993). Therefore, the first inequality in equation 3.7 holds by equation 3.6 since matrices H and S are positive semidefinite.

On the other hand, $\forall z \in R^{m+n}$, we have from lemma 3ii that

$$\varphi(z) \leq \varphi(z^*) + (z - z^*)^T \nabla\varphi(z^*) + \frac{1}{2}\lambda_{\max}(W)\|z - z^*\|^2.$$

Thus,

$$\begin{aligned} G(z, z^*) &\leq \frac{1}{2}\big[\lambda_{\max}(H)\|x - x^*\|^2 + 3\lambda_{\max}(S)\|y - y^*\|^2 \\ &\qquad + (1 + \lambda_{\max}(W))\|z - z^*\|^2\big] \\ &\leq \frac{\mu_1}{2}\|z - z^*\|^2, \quad \forall z \in R^{m+n}. \end{aligned}$$

ii. From equations 2.3, 2.5, 3.5, and 3.6, we have

$$\nabla G(z, z^*) = \begin{pmatrix} (I_m + H)(x - x^*) - Q(v - y^*) \\ (I_n + 3S)(y - y^*) + (I_n - S)(v - y^*) \end{pmatrix}.$$

Then $\forall z \in R^{m+n}$; it is straightforward to have

$$
\begin{aligned}
\nabla G(z, z^*)^T F(z) = & \, 2(x-u)^T[(I_m + H)(x-x^*) - Q(v-y^*)] \\
& + (y-v)^T[2(I_n + S)(y-y^*) - (I_n - S)(y-v)] \\
= & \, 2(u-x^*)^T[x-u-H(x-x^*) + Q(v-y^*)] \\
& + 2(x-x^*)^T H(x-x^*) + 2\|x-u\|^2 + \|y-v\|^2 \\
& + (y-v)^T S(y-v) + 2(y-y^*)^T S(y-y^*) \\
& + 2(v-y^*)^T[y-v-S(y-y^*) - Q^T(x-x^*)] \\
= & \, 2\|x-u\|^2 + \|y-v\|^2 + 2(x-x^*)^T H(x-x^*) \\
& + (y-v)^T S(y-v) + 2(y-y^*)^T S(y-y^*) \\
& + 2(u-x^*)^T(x-u-Hx-h+Qv) \\
& + 2(u-x^*)^T(Hx^* + h - Qy^*) \\
& + 2(v-y^*)^T(y-v-Sy-s-Q^T x) \\
& + 2(v-y^*)^T(Sy^* + s + Q^T x^*) \\
\geq & \, \frac{1}{2}\|F(z)\|^2 + 2(x-x^*)^T H(x-x^*) \\
& + 2(y-y^*)^T S(y-y^*) + (y-v)^T S(y-v), \qquad (3.8)
\end{aligned}
$$

where the last step follows from equation 2.1 ($u \in U$, $v \in V$),

$$(x - Hx - h + Qv - u)^T(u - x^*) \geq 0,$$

by setting $w = x - Hx - h + Qv$ and $p = x^* \in U$ in equation 1.8, and

$$(y - Sy - s - Q^T x - v)^T(v - y^*) \geq 0$$

by setting $w = y - Sy - s - Q^T x$ and $p = y^* \in V$ in equation 1.8, respectively. Therefore, lemma 4ii holds by equation 3.8 since matrices H and S are positive semidefinite.

iii. If $\min\{\lambda_{\min}(H), \lambda_{\min}(S)\} = 0$, then $\mu_2 = 0$ from the definition of μ_2, and this result can be obtained by lemma 4ii.

If $\min\{\lambda_{\min}(H), \lambda_{\min}(S)\} > 0$, then $\mu_2 > 0$. From equation 3.8 and the right-hand side of equation 3.7, we have

$$
\begin{aligned}
\nabla G(z, z^*)^T F(z) & \geq 2[\lambda_{\min}(H)\|x-x^*\|^2 + \lambda_{\min}(S)]\|y-y^*\|^2 \\
& \geq 2\min\{\lambda_{\min}(H), \lambda_{\min}(S)\}\|z-z^*\|^2 \\
& \geq 2\mu_2 G(z, z^*), \quad \forall z \in R^{m+n}.
\end{aligned}
$$

The results in lemma 4 are very important and pave the way for the stability results of neural network 2.4. In particular, neural network 2.4 has the following basic property:

Theorem 2. *For any $z^0 \in R^{m+n}$, there exists a unique and continuous solution $z(t)$ of neural network 2.4 for all $t \geq 0$ with $z(0) = z^0$.*

Proof. From equation 3.3, we have for any closed convex set Ω

$$\|P_\Omega(p) - P_\Omega(w)\| \leq \|p - w\|, \qquad \forall p, w \in R^n.$$

Thus, for any $z, z' \in R^{m+n}$, by equation 2.5 and the above inequality, we have

$$\begin{aligned}\|u - u'\| &\leq \|(I_m - H)(x - x') + Q(v - v')\| \\ &\leq \|I_m - H\| \cdot \|x - x'\| + \|Q\| \cdot \|v - v'\|\end{aligned}$$

and

$$\|v - v'\| \leq \|I_n - S\| \cdot \|y - y'\| + \|Q\| \cdot \|x - x'\|,$$

where $v' = P_V(y' - Sy' - s - Q^T x')$ and $u' = P_U(x' - Hx' - h + Qv')$. From the above two inequalities and

$$\begin{aligned}\|F(z) - F(z')\| \leq\ & 2(\|x - x'\| + \|u - u'\|) \\ & + \|y - y'\| + \|v - v'\|, \quad \forall z, z \in R^{m+n},\end{aligned}$$

we can see that $F(z)$ is Lipschitz continuous on R^{m+n}. Thus, the result can be established from theorem 1 in Han et al. (2001).

The results of lemma 2 and theorem 2 indicate that neural network model 2.4 is well defined. Now we are in the position to prove the following stability results for this model.

Theorem 3. *Neural network 2.4 is stable in the sense of Lyapunov, and for any $z^0 \in R^{m+n}$, its trajectory will reach a saddle point of $f(x, y)$ within a finite time when the scaling parameter λ is large enough. In particular, if problem 1.1 has a unique solution, then neural network 2.4 is globally asymptotically stable.*

Proof. From theorem 2, $\forall z^0 \in R^{m+n}$, let $z(t)$ be the unique and continuous solution of neural network 2.4 for all $t \geq 0$ with $z(0) = z^0$.

For the function $G(z, z^*)$ defined in equation 3.6, we have from lemma 4ii that

$$\frac{d}{dt}G(z(t), z^*) = -\lambda \nabla G(z(t), z^*)^T F(z(t)) \leq -\frac{\lambda}{2}\|F[z(t)]\|^2 \leq 0, \quad \forall t \geq 0. \tag{3.9}$$

From equation 3.9, we know that $G(z(t), z^*)$ is monotonically nonincreasing on $[0, +\infty)$. Then

$$\|z(t) - z^*\|^2 \leq 2G(z(t), z^*) \leq 2G(z^0, z^*), \qquad \forall t \geq 0$$

from the first inequality in equation 3.7. Thus, the set $\{z(t) \mid t \geq 0\}$ is bounded. Therefore, there exist a limit point $\hat{z}$ and a sequence $\{t_n\}$ with $0 < t_1 < t_2 < \ldots < t_n < t_{n+1} < \ldots$ and $t_n \to +\infty$ as $n \to +\infty$ such that

$$\lim_{k \to +\infty} z(t_k) = \hat{z}. \tag{3.10}$$

On the other hand, $\forall s \geq 0$, we have from equation 3.9 that

$$\frac{\lambda}{2}\int_0^s \|F[z(t)]\|^2 dt \leq G(z^0, z^*) - G(z(s), z^*) \leq G(z^0, z^*).$$

Thus,

$$\int_0^{+\infty} \|F[z(t)]\|^2 dt \leq \frac{2}{\lambda}G(z^0, z^*) < +\infty.$$

This implies that $\lim_{t \to +\infty} \|F[z(t)]\| = 0$. Therefore, $F(\hat{z}) = 0$ from equation 3.10, so $\hat{z} \in K^*$ from lemma 2.

By replacing $\hat{z}$ with z^* in $G(z, z^*)$, we can prove that $G(z, \hat{z}) \geq \|z - \hat{z}\|^2/2$ for all $z \in R^{m+n}$ and $G(z(t), \hat{z})$ is monotonically nonincreasing on $[0, +\infty)$. From the continuity of $G(z, \hat{z})$, it follows that $\forall \varepsilon > 0$, $\exists\, \delta > 0$ such that

$$G(z, \hat{z}) < \frac{1}{2}\varepsilon^2, \qquad \text{if } \|z - \hat{z}\| \leq \delta. \tag{3.11}$$

From equations 3.7, 3.10, and 3.11, there exists a natural number N such that

$$\|z(t) - \hat{z}\|^2 \leq 2G([z(t), \hat{z}] \leq 2G(z(t_N), \hat{z}) < \varepsilon^2, \quad \text{when } t \geq t_N.$$

That is, $\lim_{t\to+\infty} z(t) = \hat{z}$. This indicates that the solution $z(t)$ of neural network 2.4 converges to a point in K^*; that is, the solution $z(t)$ of neural network 2.4 converges to a saddle point of $f(x, y)$.

Now we show that the convergence time is finite. Without loss of generality, we let $\lim\limits_{t\to+\infty} z(t) = z^* \in K^*$. If $z^0 \notin K^*$, then $F(z^0) \neq 0$ from lemma 2. Thus, there exist $\tau > 0$ and $\delta > 0$ such that $\|F[z(t)]\| \geq \delta$ for all $t \in [0, \tau)$. It follows from equation 3.9 that

$$G[z(t), z^*] \leq G(z^0, z^*) - \frac{\lambda}{2}\int_0^t \|F[z(s)]\|^2 ds \leq G(z^0, z^*) - \frac{\lambda\delta\tau}{2}, \quad \forall t \geq \tau.$$

Therefore,

$$\|z(t) - z^*\|^2 \leq \mu_1\|z^0 - z^*\|^2 - \lambda\delta\tau, \quad \forall t \geq \tau$$

from equation 3.7. Let $\lambda = \mu_1\|z^0 - z^*\|^2/(\delta\tau)$ in the above inequality; then $\|z(t) - z^*\| \equiv 0$ for all $t \geq \tau$. This implies that $z(t) \equiv z^*$ for all $t \geq \tau$.

In particular, if problem 1.1 has a unique solution z^*, then $K^* = \{z^*\}$ since $K^* \neq \emptyset$, and for each $z^0 \in R^{m+n}$, the trajectory $z(t)$ with $z(0) = z^0$ will approach z^*. So neural network 2.4 is globally asymptotically stable at z^*.

Remark 3. Compared with the existing finite-time convergence result for model 1.4 in Xia and Feng (2004) (see theorem 3 in Xia & Feng, 2004), theorem 3 for neural network 2.4 does not require the additional condition that the initial point z^0 satisfies $(z^0 - z^*)^T M(z^0 - z^*) \neq 0$ or $[e(z^0)]^T Me(z^0) \neq 0$, where $z^* \in K^*$ and $e(z)$ is defined in equation 2.8. Unlike the existing finite-time convergence result for model 1.4 in Xia (2004) (see remark 1 in Xia, 2004), theorem 3 holds without requiring the positive definiteness of matrix H or S (see examples 3–5 in section 4).

When matrices H and S are positive definite, we have the following exponential stability result for neural network 2.4.

Theorem 4. *If matrices H and S are positive definite, then neural network 2.4 is globally exponentially stable at the unique saddle point of $f(x, y)$.*

Proof. From the hypothesis of this theorem and $K^* \neq \emptyset$, there exists a unique saddle point $z^* \in K^*$ for $f(x, y)$. From theorem 2, let $z(t)$ be the unique solution of system 2.4 with $z(0) = z^0 \in R^{m+n}$ for all $t \geq 0$.

Since matrices H and S are positive definite, we have $\lambda_{\min}(H) > 0$ and $\lambda_{\min}(S) > 0$. Thus, $\mu_2 > 0$. For the function $G(z, z^*)$ defined in equation 3.6, it follows from lemma 4iii that

$$\frac{d}{dt}G(z(t), z^*) \leq -2\lambda\mu_2 G(z(t), z^*), \quad \forall t \geq 0.$$

Thus,

$$G(z(t), z^*) \leq G(z^0, z^*)e^{-2\lambda\mu_2 t}, \quad \forall t \geq 0.$$

This and lemma 4i imply that

$$\|z(t) - z^*\| \leq \sqrt{2G(z^0, z^*)}e^{-\lambda\mu_2 t} \leq \sqrt{\mu_1}\|z^0 - z^*\|e^{-\lambda\mu_2 t}, \quad \forall t \geq 0.$$

Remark 4. Obviously, model 2.4 can be applied to solve a class of linear variational inequality problem LVI(M, q, C) defined in equation 1.3 with M, q, and C being defined in equation 2.2 and H and S being symmetric (see examples 2 and 5 in section 4).

Remark 5. The results obtained in this article hold for any closed convex subset of R^n. In particular, the common cases for U or V are (1) $\{x \in R^n | \|x\| \leq c\}$ (ball or l_2 norm constraint) and (2) $\{x \in R^n | \sum_{i=1}^n |x_i| \leq d\}$ (l_1 norm constraint).

4 Illustrative Examples

In this section, five examples are provided to illustrate the theoretical results achieved in section 3 and the simulation results of the dynamical system 2.4. The simulation is conducted in Matlab, and the ODE solver used is ODE45 which is a nonstiff medium-order method.

Example 1 shows the effectiveness of the proposed neural network 2.4 for problem 1.1 with infinite number of saddle points.

4.1 Example 1. Consider problem 1.1 with $U = V = \{x \in R^2 \, |x \geq 0\}$, $h = s = (0, 0)^T$,

$$H = \begin{pmatrix} 4 & -2 \\ -2 & 1 \end{pmatrix}, \quad Q = \begin{pmatrix} -1 & 1 \\ 1 & -2 \end{pmatrix}, \quad \text{and} \quad S = \begin{pmatrix} 1 & -2 \\ -2 & 4 \end{pmatrix}.$$

This problem has an infinite number of saddle points $x^* = (0, 0)^T$ and $y^* = (2r, r)^T$ $(r \geq 0)$.

From the analysis in section 3, we use neural network 2.4 to solve this problem; all simulation results show that neural network 2.4 always converges to one of its equilibrium points. For example, let $\lambda = 100$. Figure 2 shows the convergence behavior of the error $\|z(t) - z^*\|$ based on neural network 2.4 with 20 random initial points.

Example 2 shows that the proposed neural network 2.4 can be applied to solve a class of linear variational inequality problems.

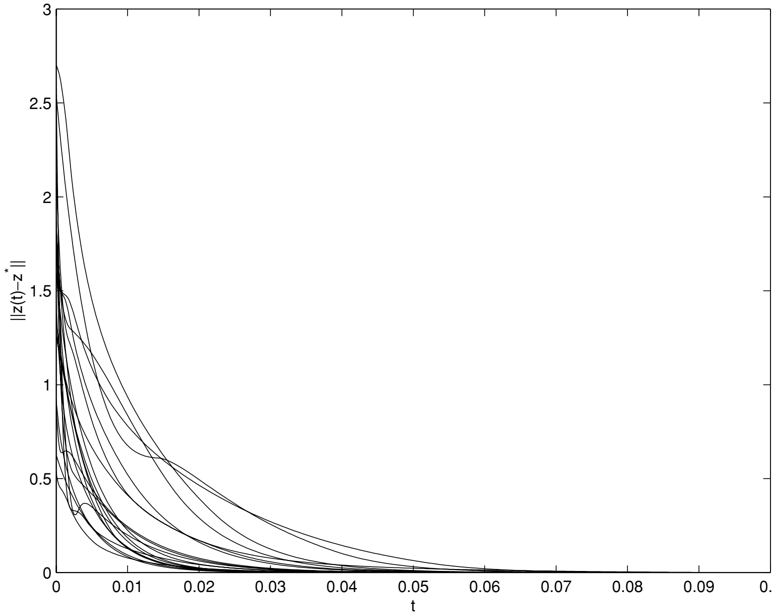


Figure 2: Convergence behavior of the error $\|z(t) - z^*\|$ based on neural network 2.4 with 20 random initial points in example 1.

4.2 Example 2. Consider the linear variational inequality problem LVI(M, q, C) defined in equation 1.3 with $C = \{z \in R^4 \mid -8 \le z_i \le 9, i = 1, 2, 3, 4\}$,

$$M = \begin{pmatrix} 0.1 & 0.1 & 0.5 & -0.5 \\ 0.1 & 0.1 & -0.5 & 0.5 \\ -0.5 & 0.5 & 0.2 & 0.1 \\ 0.5 & -0.5 & 0.1 & 0.05 \end{pmatrix}, \quad \text{and} \quad q = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}.$$

This problem has a unique solution $z^* = (1, -1, -2/3, 4/3)^T$.

Let $U = V = \{x \in R^2 \mid -8 \le x_i \le 9,\ i = 1, 2\}$, $h = s = (1, -1)^T$,

$$H = \begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}, \quad Q = \begin{pmatrix} -0.5 & 0.5 \\ 0.5 & -0.5 \end{pmatrix}, \quad \text{and} \quad S = \begin{pmatrix} 0.2 & 0.1 \\ 0.1 & 0.05 \end{pmatrix}.$$

Then model 2.4 can be applied to solve this problem from remark 4. All our simulation results show that neural network 2.4 is asymptotically stable at z^*. For example, let $\lambda = 100$. Figure 3 displays the convergence behavior

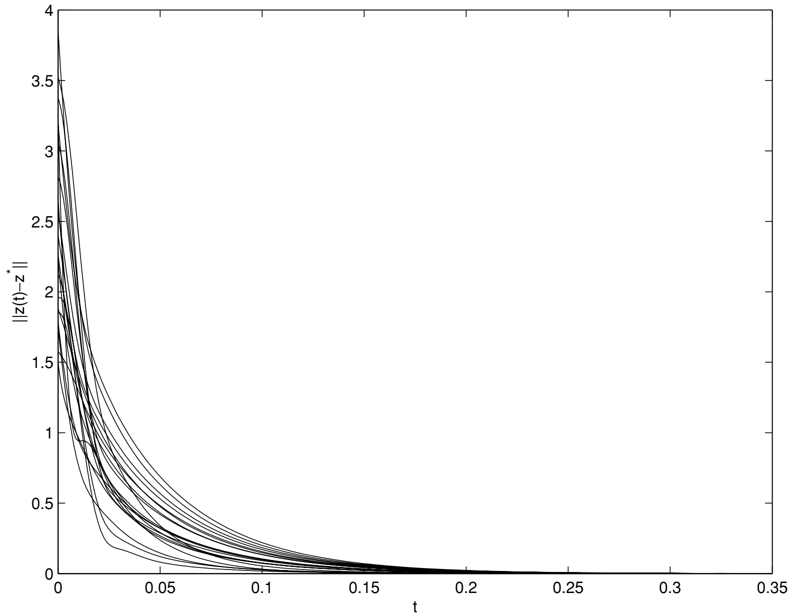


Figure 3: Convergence behavior of the error $\|z(t) - z^*\|$ based on neural network 2.4 with 20 random initial points in example 2.

of the error $\|z(t) - z^*\|$ based on neural network 2.4 with 20 random initial points.

It should be mentioned that neural network 1.4 cannot be used to solve this problem. In fact, Figure 4 shows model 1.4 with initial point $(2, 2, 2, 2)^T \in R^4$ and $\lambda = 100$ is not stable, where the error $\|z(t) - z^*\|$ approaches 0.0178632.

Example 3 shows that the proposed neural network 2.4 can be applied to solve large-scale problems.

4.3 Example 3. Consider problem 1.1 with $U = \{x \in R^{2n} \mid -1 \le x \le 1\}$, $V = \{y \in R^n \mid -1 \le y \le 1\}$, $h = (0, 0, \dots, 0)^T \in R^{2n}$, $s = -(1, 1, \dots, 1)^T \in R^n$, S being an $n \times n$ zero matrix,

$$H = \begin{pmatrix} 1 & -1 & 0 & \dots & 0 & 0 \\ -1 & 2 & -1 & \dots & 0 & 0 \\ 0 & -1 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2 & -1 \\ 0 & 0 & 0 & \dots & -1 & 1 \end{pmatrix}_{2n\times 2n}, \quad \text{and} \quad Q = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 \\ 0 & 0 & \dots & 1 \end{pmatrix}_{2n\times n}.$$

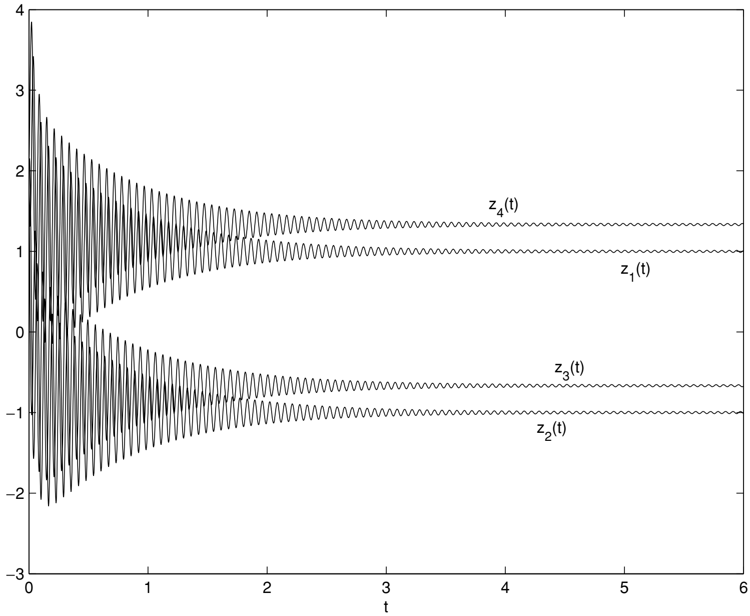


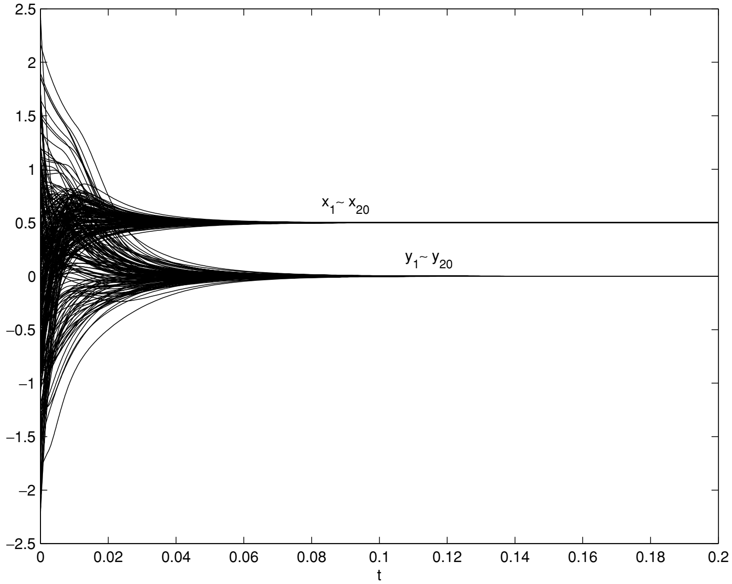
Figure 4: Transient behavior of model 1.4 in example 2.

This problem has a unique saddle point $x^* = (0.5, 0.5, \ldots, 0.5)^T \in R^{2n}$ and $y^* = (0, 0, \ldots, 0)^T \in R^n$.

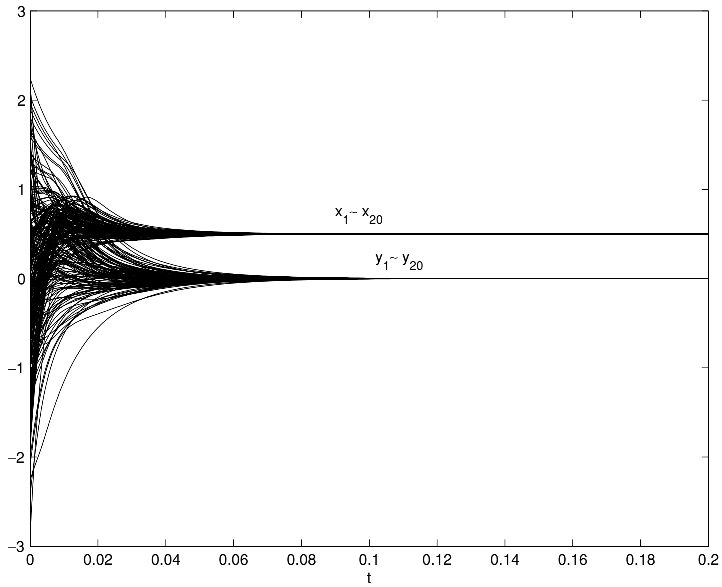
We use neural network 2.4 to solve this problem; all simulation results show that this neural network is asymptotically stable at z^*. For example, let $\lambda = 100$. Figures 5a and 5b show the trajectories of the first 20 components of x and y of neural network 2.4 with 6 random initial points z^0 for $n = 1500$ and 2000, respectively.

Next, we compare the proposed model 2.4 with other methods. For simplicity, we let $N = I_{m+n}$ in method 2.10. Figure 6 shows that model 1.4 with initial point $(0, 0, \ldots, 0)^T \in R^{18}$ and $\lambda = 10$ is not stable, where the error $\|z(t) - z^*\|$ approaches 1.0327. Tables 1 and 2 report the numerical results with two different initial points obtained by methods 2.4, 2.7, 2.9, 2.10, and 2.11, respectively, where "Iter". represents the iterative number; t_f denotes the time that the stopping criterion $\|\frac{dz}{dt}/\lambda\| < 10^{-5}$ is met for models 2.4, 2.7, and 2.9; and the stopping rule for methods 2.10 and 2.11 is $\|z^{k+1} - z^k\| < 10^{-5}$. From Tables 1 and 2, we can see that the proposed method not only provides a better solution but also has a faster convergence than methods 2.7, 2.9, 2.10, and 2.11, except for method 2.10 with $\theta = 1$.

Example 4 illustrates that the proposed neural network 2.4 has a faster convergence than other methods.



(a) $n = 1500$



(b) $n = 2000$

Figure 5: Transient behavior of model 2.4 in example 3.

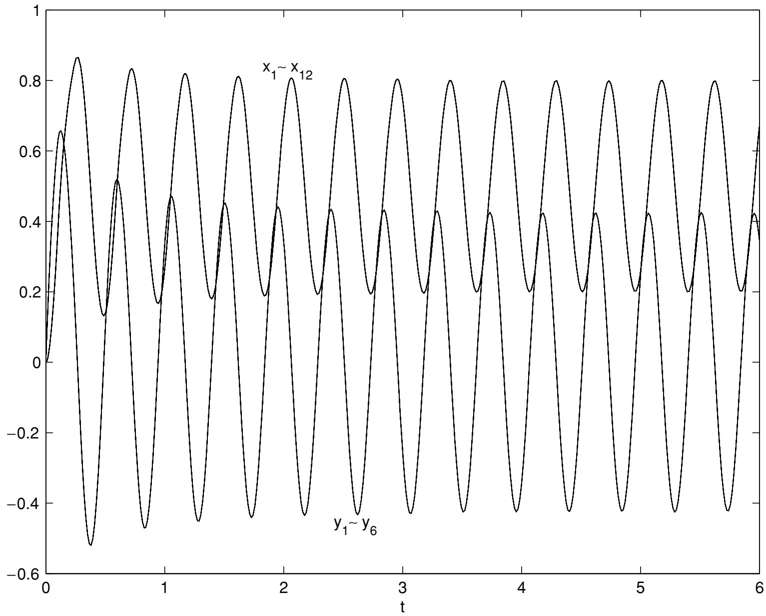


Figure 6: Transient behavior of model 1.4 in example 3.

Table 1: Numerical Results of Example 3 with Initial Point $-(1, 1, \ldots, 1) \in R^{1800}$.

Method	Parameters	Iter.	CPU (sec.)	$\|z(t_f) - z^*\|$
2.4	$\lambda = 100$, $t_f = 0.0931$	81	3.110	5.34×10^{-6}
2.7	$\lambda = 100$, $t_f = 0.1803$	121	9.641	4.08×10^{-6}
2.9	$\lambda = 100$, $\theta = 1.8$, $t_f = 0.2631$	117	8.797	6.80×10^{-6}
	$\lambda = 100$, $\theta = 1$, $t_f = 0.2516$	89	6.672	1.22×10^{-5}
2.10	$\theta = 1.8$	106	3.578	6.35×10^{-6}
	$\theta = 1$ (best θ value)	28	0.985	1.09×10^{-5}
	$\theta = 0.2$	105	3.563	5.81×10^{-5}
2.11	$\theta = 0.2475$, $\nu = 0.99$	246	8.291	2.43×10^{-5}
	$\theta = 0.15$, $\nu = 0.6$	604	20.172	4.46×10^{-5}

Example 4. Consider problem 1.1 with $U = \{x \in R^4 | x_i \geq 0, i = 1, \ldots, 4\}$, $V = \{y \in R^4 | y_i \geq 0, i = 1, 2\}$, H and S are 4×4 zero matrix, $h = -(6, 6, 5, 5)^T$, $s = -(0, 0, 10, 5)^T$, and

$$Q = \begin{pmatrix} 1 & -2 & 1 & 0 \\ -1 & 30 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Table 2: Numerical Results of Example 3 with Initial Point $(2, 2, \ldots, 2) \in R^{1800}$.

Method	Parameters	Iter.	CPU (sec.)	$\|z(t_f) - z^*\|$
2.4	$\lambda = 100, t_f = 0.0949$	73	3.062	5.33×10^{-6}
2.7	$\lambda = 100, t_f = 0.1559$	113	8.203	4.08×10^{-6}
2.9	$\lambda = 100, \theta = 1.8, t_f = 0.1568$	85	6.968	6.80×10^{-6}
	$\lambda = 100, \theta = 1, t_f = 0.2599$	89	6.453	1.22×10^{-5}
2.10	$\theta = 1.8$	109	3.688	6.12×10^{-6}
	$\theta = 1$ (best θ value)	30	1.047	7.98×10^{-6}
	$\theta = 0.2$	113	3.828	5.76×10^{-5}
2.11	$\theta = 0.2475, \nu = 0.99$	240	8.032	2.42×10^{-5}
	$\theta = 0.15, \nu = 0.6$	591	19.766	4.51×10^{-5}

Then this problem has a unique solution $x^* = (10, 5, 0, 0)^T$ and $y^* = (0, 0, -6, -6)^T$.

From the analysis in section 3, neural network 2.4 can be applied to solve this problem. In this case, it becomes

$$\frac{dz}{dt} = \frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\lambda \begin{pmatrix} 2\{x - P_U[x - h + QP_V(y - Q^T x - s)]\} \\ y - P_V(y - Q^T x - s) \end{pmatrix}, \tag{4.1}$$

where $x \in R^4$ and $y \in R^4$. All simulation results show that neural network 4.1 is always asymptotically stable at (x^*, y^*). For example, let $\lambda = 1000$. Figure 7 shows that the convergence behavior of the error $\|z(t) - z^*\|$ based on equation 4.1 with 20 random initial points.

Next, we compare the proposed model 2.4 with other methods. For simplicity, we let $N = I_{m+n}$ in method 2.10. As a matter of fact, Figure 8 shows that model 1.4 with initial point $(2, 2, \ldots, 2)^T \in R^8$ and $\lambda = 100$ is not stable, where the error $\|z(t) - z^*\|$ approaches 1.38113. Tables 3 and 4 report the numerical results obtained by methods 2.4, 2.7, 2.9, 2.10, and 2.11 with two different initial points, respectively, where Iter., t_f, and the stopping criterion are the same as for example 3. From Tables 3 and 4, we can see that the proposed method not only provides a better solution but also has a faster convergence than other methods.

Example 5 shows that the conditions of theorem 3 in Xia and Feng (2004) are not enough to ensure the finite-time convergence of model 1.4.

Example 5. Consider the linear variational inequality problem LVI(M, q, C) defined in equation 1.3 with $C = R^3$:

$$M = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ 1 & 1 & 0 \end{pmatrix}, \quad \text{and} \quad q = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

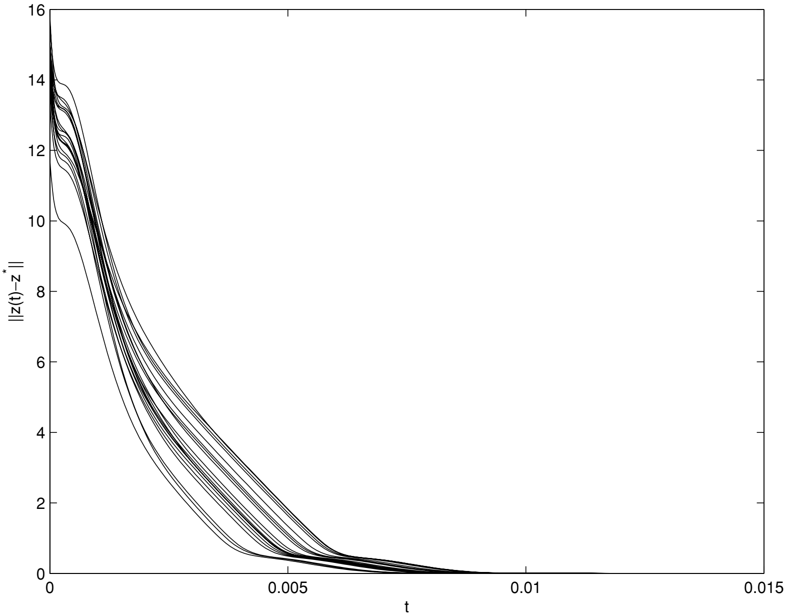


Figure 7: Convergence behavior of the error $\|z(t) - z^*\|$ based on equation 4.1 with 20 random initial points in example 4.

Table 3: Numerical Results of Example 4 with Initial Point $-(2, 2, \ldots, 2)^T \in R^8$.

Method	Parameters	Iter.	CPU (sec.)	$\|z(t_f) - z^*\|$
2.4	$\lambda = 1000$, $t_f = 0.0187$	177	0.062	3.45×10^{-6}
2.7	$\lambda = 1000$, $t_f = 0.0490$	53,585	13.829	5.98×10^{-6}
2.9	$\lambda = 1000, \theta = 1.8, t_f = 0.4575$	745	0.312	1.60×10^{-5}
	$\lambda = 1000, \theta = 1, t_f = 0.9082$	1281	0.516	4.03×10^{-5}
2.10	$\theta = 1.8$	19,229	1.297	1.25×10^{-4}
	$\theta = 0.9$ (best θ value)	2290	0.171	4.76×10^{-4}
	$\theta = 0.2$	4140	0.266	1.69×10^{-3}
2.11	$\theta = 0.0329$, $\nu = 0.99$	20,661	1.656	3.04×10^{-4}
	$\theta = 0.0199$, $\nu = 0.6$	45,813	3.672	5.01×10^{-4}

This problem has a unique solution $z^* = (-0.5, -0.5, 0)^T$.

From remark 4, this problem can be solved by model 2.4. In this case, model 2.4 becomes

$$\frac{dz}{dt} = \frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\lambda \begin{pmatrix} 2[Hx - Q(y - Q^T x - s)] \\ Q^T x + s \end{pmatrix}, \tag{4.2}$$

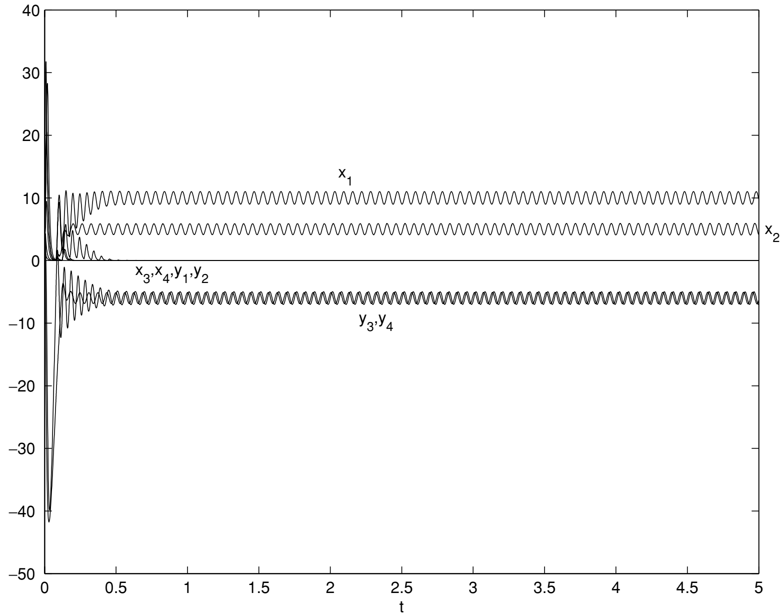


Figure 8: Transient behavior of model 1.4 in example 4.

Table 4: Numerical Results of Example 4 with Initial Point $(2, 2, \ldots, 2)^T \in R^8$.

Method	Parameters	Iter.	CPU (sec.)	$\|z(t_f) - z^*\|$
2.4	$\lambda = 1000$, $t_f = 0.0190$	141	0.047	5.99×10^{-6}
2.7	$\lambda = 1000$, $t_f = 0.0539$	58,965	13.563	7.61×10^{-6}
2.9	$\lambda = 1000$, $\theta = 1.8$, $t_f = 0.5038$	785	0.312	1.22×10^{-5}
	$\lambda = 1000$, $\theta = 1$, $t_f = 0.8924$	1245	0.500	4.41×10^{-5}
2.10	$\theta = 1.8$	19,026	1.234	1.26×10^{-4}
	$\theta = 0.9$ (best θ value)	2319	0.156	3.85×10^{-4}
	$\theta = 0.2$	5292	0.360	1.39×10^{-3}
2.11	$\theta = 0.0329$, $\nu = 0.99$	20,746	1.671	3.04×10^{-4}
	$\theta = 0.0199$, $\nu = 0.6$	46,007	3.719	5.02×10^{-4}

where $x \in R^2$, $y \in R^1$, $s = 1$,

$$H = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \quad \text{and} \quad Q = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

All simulation results show that neural network 4.2 is always asymptotically stable at z^*. For example, let $\lambda = 100$. Figure 9 displays the convergence

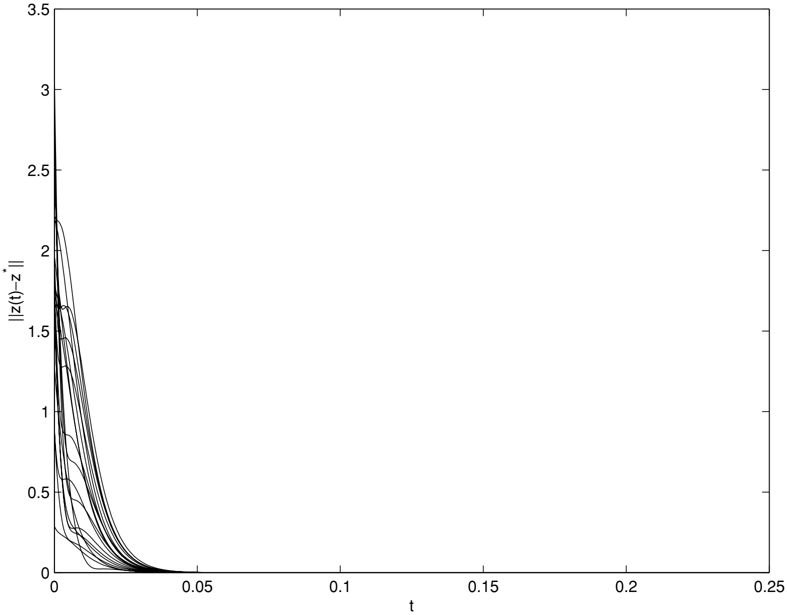


Figure 9: The convergence behavior of the error $\|z(t) - z^*\|$ based on equation 4.2 with 20 random initial points in example 5.

behavior of the error $\|z(t) - z^*\|$ based on equation 4.2 with 20 random initial points.

It should be mentioned that model 1.4 cannot be used to solve this problem. When applied to this problem, model 1.4 becomes

$$\begin{cases} \frac{dz_1}{dt} = \lambda(z_2 - z_1 + z_3), \\ \frac{dz_2}{dt} = \lambda(z_1 - z_2 + z_3), \\ \frac{dz_3}{dt} = -\lambda(z_1 + z_2 + 1). \end{cases} \tag{4.3}$$

It is easy to verify that the solution of equation 4.3 is

$$\begin{cases} z_1(t) = \left[\sqrt{2}z_3^0\sin(\omega_1 t) - \sqrt{2}\omega_2\cos(\omega_1 t) - 1 + \left(z_1^0 - z_2^0\right)e^{-2\lambda t}\right]/2, \\ z_2(t) = \left[\sqrt{2}z_3^0\sin(\omega_1 t) - \sqrt{2}\omega_2\cos(\omega_1 t) - 1 - \left(z_1^0 - z_2^0\right)e^{-2\lambda t}\right]/2, \\ z_3(t) = z_3^0\cos(\omega_1 t) + \omega_2\sin(\omega_1 t), \end{cases}$$

where $\omega_1 = \sqrt{2}\lambda$ and $\omega_2 = -\sqrt{2}(z_1^0 + z_2^0 + 1)/2$. Obviously equation 4.3 is divergent and has no finite-time convergence when $|\omega_2| + |z_3^0| > 0$. However, for any $z^0 \in R^3$ with $|\omega_2| + |z_3^0| > 0$ and $(z^0 - z^*)^T M(z^0 - z^*) + [e(z^0)]^T Me(z^0) > 0$ ($e(z)$ is defined in equation 2.8), for example, $z^0 = (-3, 0, 0)^T$, the conditions of theorem 3 in Xia and Feng (2004) are satisfied. Thus, the conditions of theorem 3 in Xia and Feng (2004) are not enough to ensure the finite-time convergence of model 1.4 when the model's trajectory $z(t)$ with $z(0) = z^0 \in U \times V$ does not converge to one of its equilibrium point z^*.

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

Remark 6. (i) For model 1.4 the simulation results show that it is stable for example 1, yet unlike model 2.4, its stability and convergence might not be guaranteed when initial point $z^0 \notin U \times V$ (see Friesz et al, 1994; Gao, 2003, 2004; Gao et al., 2004; Xia, 2004; Xia & Feng, 2004; Xia & Wang, 2001), even the matrices H and S are positive semidefinite (see examples 2–5). (ii) For method 2.10, computational results for example 3 show that method 2.10 is better than others when $\theta = 1$, yet unlike model 2.4, it is not suitable for parallel implementation due to the choice of the varying parameter $\gamma(z)$ as mentioned in model 2.9, and its performance depends on the choices of parameters N and θ. (iii) Since H is positive semidefinite and $S = 0$ in examples 3 to 5, the existing finite-time convergence result for model 1.4 in Xia (2004) cannot be applied to these examples (see remark 1 in Xia, 2004).

5 Conclusion

In this letter, we have proposed a new neural network for solving a class of convex quadratic minimax problems by means of its inherent properties. We have shown that the new model is stable in the sense of Lyapunov and converges to an exact saddle point in finite time when matrices H and S are positive semidefinite. Furthermore, the global exponential stability of the proposed neural network is also obtained under certain conditions. Compared with the existing neural networks and two typically numerical methods, the proposed neural network has finite-time convergence, a simpler structure, and lower complexity. Thus, the proposed neural network is more suitable for hardware implementation. Since the new network can be applied directly to solve a class of linear variational inequality problems and a broad set of classes of optimization problems, it has great application potential. Illustrative examples confirm the theoretical results and demonstrate that our new model is reliable and attractive.

Acknowledgments

We are very grateful to the two anonymous reviewers for their comments and constructive suggestions on earlier versions of this article. The research was supported in part by grants from Hong Kong Baptist University, the Research Grant Council of Hong Kong, and NSFC Grant of China No. 10471083.

References

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (1993). *Nonlinear programming theory and algorithms* (2nd ed.). New York: Wiley.

Bouzerdorm, A., & Pattison, T. R. (1993). Neural network for quadratic optimization with bound constraints. *IEEE Trans. Neural Networks, 4*, 293–304.

Friesz, T. L., Bernstein, D. H., Mehta, N. J., Tobin, R. L., & Ganjlizadeh, S. (1994). Day-to-day dynamic network disequilibria and idealized traveler information systems. *Operations Research, 42*, 1120–1136.

Fukushima, M. (1992). Equivalent differentiable optimization problems and descent method for asymmetric variational inequality problems. *Mathematical Programming, 53*, 99–110.

Gao, X. B. (2003). Exponential stability of globally projected dynamic systems. *IEEE Trans. Neural Networks, 14*, 426–431.

Gao X. B. (2004). A novel neural network for nonlinear convex programming. *IEEE Trans. Neural Networks, 15*, 613–621.

Gao, X. B., & Liao, L.-Z. (2003). A neural network for monotone variational inequalities with linear constraints. *Physics Letters A, 307*, 118–128.

Gao, X. B., Liao, L.-Z., & Xue, W. M. (2004). A neural network for a class of convex quadratic minimax problems with constraints. *IEEE Trans. Neural Networks, 15*, 622–628.

Han, Q. M., Liao, L.-Z., Qi, H. D., & Qi, L. Q. (2001). Stability analysis of gradient-based neural networks for optimization problems. *J. Global Optim., 19*, 363–381.

He, B. S. (1996). Solution and application of a class of general linear variational inequalities. *Science in China, series A, 39*, 395–404.

He, B. S. (1999). Inexact implicit methods for monotone general variational inequalities. *Mathematical Programming, 86*, 199–217.

He, B. S., & Yang, H. (2000). A neural-network model for monotone linear asymmetric variational inequalities. *IEEE Trans. Neural Networks, 11*, 3–16.

Kinderlehrer, D., & Stampacchia, G. (1980). *An introduction to variational inequalities and their applications*. New York: Academic Press.

Ortega, T. M., & Rheinboldt, W. C. (1970). *Iterative solution of nonlinear equation in several variables*. New York: Academic Press.

Rockafellar, R. T. (1987). Linear-quadratic programming and optimal control. *SIAM J. Control Optim., 25*, 781–814.

Smith, T. E., Friesz, T. L., Bernstein, D. H., & Suo, Z. G. (1997). A comparative analysis of two minimum-norm projective dynamics and their relationship to variational inequalities. In M. C. Ferris, J. S. Pang (Eds.), *Complementarity and variational problems: State of art* (pp. 405–424). Philadelphia: SIAM.

Solodov, M. V., & Tseng, P. (1996). Modified projection-type methods for monotone variational inequalities. *SIAM J. Control Optim., 27*, 1814–830.

Tseng, P. (2000). A modified forward-backward splitting method for maximal monotone mappings. *SIAM J. Control Optim., 38*, 431–446.

Xia, Y. S. (2004). An extended projection neural network for constrained optimization. *Neural Computation, 16*, 863–883.

Xia, Y. S., & Feng, G. (2004). On convergence rate of projection neural networks. *IEEE Trans. Automatic Control, 49*, 91–96.

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

Xia, Y. S., & Wang, J. (1998). A general methodology for designing globally convergent optimization neural networks. *IEEE Trans. Neural Networks, 9*, 1331–1343.

Xia, Y. S., & Wang, J. (2000). On the stability of globally projected dynamical systems. *J. Optim. Theory Appl., 106*, 129–160.

Xia, Y. S., & Wang, J. (2001). Global asymptotic and exponential stability of a dynamic neural system with asymmetric connection weights. *IEEE Trans. Automatic Control, 46*, 635–638.

Received August 9, 2004; accepted June 14, 2005.