Neurocomputing with Time Delay Analysis for Solving Convex Quadratic Programming Problems

Yen-Hung Chen and Shu-Cherng Fang

***Abstract*—This paper presents a neural-network computational scheme with time-delay consideration for solving convex quadratic programming problems. Based on some known results, a delay margin is explicitly determined for the stability of the neural dynamics, under which the states of the neural network does not oscillate. The configuration of the proposed neural network is provided. Operational characteristics of the neural network are demonstrated via numerical examples.**

***Index Terms*—Artificial neural network, convex programming, Hopfield network, linear and quadratic programming, time-delay dynamic system.**

I. Introduction

For computational efficiency, an analog circuit, also known as *dynamic solver* or *analog computer*, was first proposed by Dennis [9] in 1959, and further extended by Stern [34] in 1965. Later the Hopfield network [16] was proposed in 1982 and used to solve a 30-city traveling salesman problem [18] in 1985. Thereafter, numerous variations of the Hopfield network have been developed for solving optimization problems [35], [32], [39], [23], [24], [36]. However, these variations made a critical assumption that neurons communicate and respond instantaneously without any time delay. In reality, this assumption is not correct, because some hardware characteristics such as the switching delays, parameter variability, parasitic capacitance, and inductance do exist in an electronic neural network. The resulting time delay could lead to some instabilities that were not predicted by theory [7].

The dynamic behavior of the Hopfield network with time delay has been investigated in recent years. The time delay margins were derived for network stability with respect to both the low-gain and high-gain neuron activation functions by Marcus *et al.* ([25]–[27]). In [1], different time delays were considered for the analysis of neural dynamics, in particular, for oscillations. A delay-independent stability condition was derived for Hopfield networks with asymmetric connection weight matrices by Gopalsamy and He [12]. Generalized delay-independent stability conditions were given for various Hopfield networks with time delay in [40], [13], [37], and [38]. In view of control theory, the neural dynamics of a Hopfield network with time delay simply represents a system of difference-differential equations. Hence the stability can be analyzed in a much broader context [8], [2], [6], [14], [19]–[21], [28]–[31].

Notice that a neural network designed for solving optimization problems may be quite different from a regular Hopfield network. In such a neural network, its connection weight-matrix is neither symmetric nor normalized. The matrix size is determined by the number of variables and the number of constraints. The elements of the matrix are the coefficients in the constraints. Being a solution to an optimization problem, the corresponding state of the network is not necessarily bounded. Neither are the outputs of neuron activation functions guaranteed to be bounded. Consequently, the research results done for the regular Hopfield network with time delay cannot be directly applied for solving the optimization problems.

The objective of this paper is to conduct an explicit analysis of neurocomputing with time delay for solving the convex quadratic (including linear) programming (QP) problems. Building upon the results reported in [4], we first convert a convex QP problem with equality constraints into a neural network with time delay in Section II. Then a delay margin is derived for stability analysis of the network in Section III. The configuration of the proposed neural network is presented in Section IV. Numerical examples are included in Section V to demonstrate the characteristics of the network with different time delay values. Some concluding remarks are made in Section VI.

II. Problem Formulation

Consider the following convex QP problem [10]:

$$\begin{aligned}&(\mathrm{QP}) \min \frac{1}{2}x^T Qx + c^T x\\ &\text{s.t. } Ax = b, \qquad x \geq 0\end{aligned} \tag{1}$$

where $c \in R^n$, $A \in R^{m\times n}$, $b \in R^m$, $x \in R^n$, and $Q \in R^{n\times n}$ is symmetric and positive semidefinite. Assume that feasible domain

$$F = \{x \in R^n \mid Ax - b = 0 \text{ and } x \geq 0\} \tag{2}$$

is not empty and the objective function is bounded below over F. Note that, when Q is a zero matrix, (QP) becomes a linear programming problem.

To adopt the neural-network approach for solving problem (QP), we can relax the problem into an unconstrained optimization problem by using a convex penalty function $\Phi(\cdot)$ to penalize the violation of constraints. A basic requirement of the penalty function is that

$$\Phi(\cdot)\begin{cases} =0, & \text{if no violation of constraints}\\ >0, & \text{otherwise}\end{cases} \tag{3}$$

Manuscript received March 16, 1998; revised June 3, 1999 and October 19, 1999. This work was supported in part by a National Textile Center Research Grant.

Y.-H. Chen is with Product Management, i2 Technologies, Irving, TX 75039 USA (e-mail: yen_hung@yahoo.com).

S.-C. Fang is with Industrial Engineering and Operations Research, North Carolina State University, Raleigh, NC 27695-7906 USA fang@eos.ncsu.edu.

Publisher Item Identifier S 1045-9227(00)00899-7.

and $\Phi(\cdot)$ is convex and piecewise differentiable. Typical examples such as quadratic, p-norm, Huber's, and logistic penalty functions can be found in [5].

With the help of the penalty function, problem (QP) can be reformulated as follows:

$$(\mathrm{QP}_\Phi)\ \min \frac{1}{2}x^TQx + c^Tx \qquad \text{s.t.}\ \sum_{i=1}^{m}\Phi(A_ix - b_i) = 0, \qquad x \geq 0 \tag{4}$$

where A_i is the ith row of matrix A, and b_i is the ith element of vector b, for $i = 1, 2, \cdots, m$.

Let $X = \{x \in R^n \mid x \geq 0\}$. By incorporating the explicit constraints into the objective function of problem (QP_Φ), the problem can be transformed into the following unconstrained convex programming problem:

$$\min_{x\in X} E(x, s) \triangleq \min_{x\in X}\left\{\frac{1}{2}x^TQx + c^Tx + s\sum_{i=1}^{m}\Phi(A_ix - b_i)\right\} \tag{5}$$

and $s > 0$ is a penalty parameter. This provides an energy function for the neural network corresponding to problem (QP). Note that all decision variables in (QP) become the state variables in the energy function of the network. Moreover, they are time-dependent ([16], [17]), i.e., $x = x(t)$, $t \geq 0$. Our objective is to find a solution to

$$\min_{x(t)\in X} E(x(t), s) \triangleq \min_{x(t)\in X}\left\{\frac{1}{2}x(t)^TQx(t) + c^Tx(t) + s\sum_{i=1}^{m}\Phi(A_ix(t) - b_i)\right\}. \tag{6}$$

Then, based on the previous work [4], we can develop a neural network for solving the convex (QP).

III. Time-Delay Analysis

Consider a linear time-delay system described by a system of differential-difference equations of the following form:

$$\dot{x}(t) = Mx(t) + M_\tau x(t - \tau) \tag{7}$$

with $x(t_0) = \Theta(t_0)$, $t_0 \in [-\tau, 0)$, and $x(0) = \Theta_0 \in R^n$. Here $x(t) \in R^n$, $t \geq 0$, $M, M_\tau \in R^{n\times n}$, $\tau \geq 0$ is a time-independent delay, and $\Theta(\cdot)$ is an initial value function of $x(t)$. Given $\Theta(t_0) \in L^2(-\tau, 0; R^n)$, Pritchard and Salamon [33] have shown that a unique solution of system (7) exists and it is continuously dependent on the initial value function $\Theta(\cdot)$ and initial data $x(0)$. The stability of an equilibrium point of the system depends on the solution $\hat{\lambda}$ of the following characteristic equation of (7):

$$\det(\hat{\lambda}I - M - M_\tau e^{-\tau\hat{\lambda}}) = 0. \tag{8}$$

Equivalently, the solution is generally represented as an eigenvalue of the matrix $M + M_\tau e^{-\tau\hat{\lambda}}$, i.e.,

$$\hat{\lambda} = \lambda_i(M + M_\tau e^{-\tau\hat{\lambda}}) \tag{9}$$

where $\lambda_i(M + M_\tau e^{-\tau\hat{\lambda}})$ represents an eigenvalue for each i. Kamen [20] has shown that the characteristic (8) has an infinite number of solutions. Therefore, the range of index i is not finite.

There are two stability conditions reported for an equilibrium point of the system defined by (7). Let us define

$$l_1 \triangleq \mu_p(M) + \|M_\tau\|_p \tag{10}$$

and

$$l_2 \triangleq \mu_p(-\iota M) + \|M_\tau\|_p \tag{11}$$

where $\iota^2 = -1$, $p = 1, 2$, and ∞, $\mu_p(M)$ and $\mu_p(-\iota M)$ are matrix measures of M and $-\iota M$, respectively, and $\|M_\tau\|_p$ is a matrix norm of M_τ. The first stability condition is given by Mori *et al.* [30] as the following theorem.

Theorem 1: If $l_1 < 0$, every equilibrium point of the dynamic system defined by (7) is asymptotically stable.

Since the condition does not depend on the size of time delay, the condition is generally referred to as a "delay-independent stability condition" [2].

The second stability condition, which depends on the size of time delay and hence is referred to as a "delay-dependent stability condition," was given by Mori and Kokame [31] in the following theorem.

Theorem 2: When $l_1 \geq 0$, if

$$\mathrm{Re}\,\lambda_i(M + M_\tau e^{-\tau\hat{\lambda}}) < 0 \tag{12}$$

for $i = 1, 2, \ldots$, and $\hat{\lambda}$ taking the values of $\iota\omega$, $l_1 + \iota\omega$, and $r + \iota l_2$, with $0 \leq \omega \leq l_2$ and $0 \leq r \leq l_1$, where $\iota^2 = -1$ and $\mathrm{Re}\,\lambda_i(\cdot)$ represents the real part of the eigenvalue, then every equilibrium point of the dynamic system defined by (7) is asymptotically stable.

The two conditions complement each other in a way such that, by checking the stability conditions of the above two theorems, the stability conditions of an equilibrium point of the system defined by (7) can be clearly determined. In particular, consider a dynamic system described by the following difference-differential equation:

$$\dot{x}(t) = ax(t) + bx(t - \tau) \tag{13}$$

with $x(t_0) = \theta(t_0), t_0 \in [-\tau, 0)$, and $x(0) = \theta_0 \in R$, where $x(t), a, b \in R$, $t, \tau \geq 0$, and $\theta(\cdot)$ is an initial value function of $x(t)$. By Theorems 1 and 2, we can prove that every equilibrium point of the system is asymptotically stable (see [3, pp. 39–45]), if

$$a + b < 0, \qquad \text{with } b \geq 0 \tag{14}$$

$$a + be^{-\tau(a-b)}\cos(\tau b) < 0, \qquad \text{with } -\frac{\pi}{2\tau} \leq b < 0 \tag{15}$$

$$a + b\cos(\tau b) < 0, \qquad \text{with } -\frac{\pi}{\tau} \leq b < -\frac{\pi}{2\tau}, \quad \text{and} \tag{16}$$

$$a - b < 0, \qquad \text{with } b < -\frac{\pi}{\tau}. \tag{17}$$

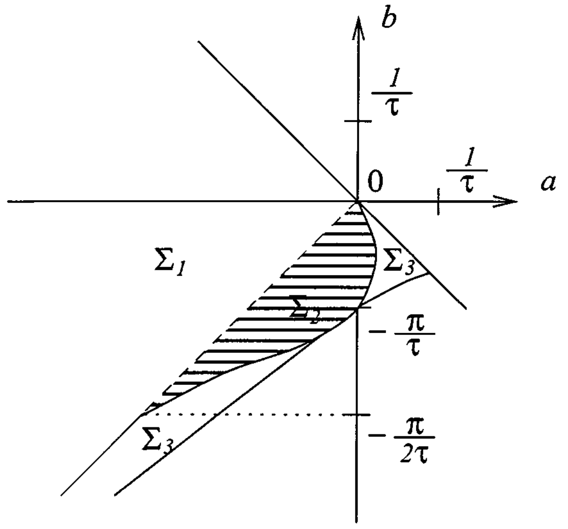


Fig. 1. Stability region in (a, b)-plane of system (13).

These inequalities define the regions $\sum_1$ and $\sum_2$ on the (a, b)-plane of Fig. 1.

For the energy function $E(x(t), s)$ defined by (6), if a time-independent delay $\tau \geq 0$ is present in the circuit implementation of the proposed neural networks, the neural dynamics of the neural networks can be described by

$$\dot{x}(t) = -\eta \nabla_{x(t)} E(x(t-\tau), s) \tag{18}$$

for $x(t_0) = \Theta(t_0), t_0 \in [-\tau, 0)$, and $x(0) = \Theta_0 \in R^n$, where $t, \tau \geq 0$, $\Theta(\cdot)$ is an initial value function of the system, $\eta > 0$ is a learning coefficient, $s > 0$ is a penalty parameter, and

$$\nabla_{x(t)} E(x(t-\tau), s) = Qx(t-\tau) + c + sA^T \phi(Ax(t-\tau) - b) \tag{19}$$

for $\phi = [\phi_i]^T$, and $\phi_i = \phi$ is a derivative of the penalty function Φ in (6), for $1 \leq i \leq m$.

When there is no time delay, i.e., $\tau = 0$, Chen and Fang [4] have shown that, as $s \to \infty$, an optimal solution of problem (QP) is an equilibrium point of the neural dynamics (18), and vise versa. Furthermore, as $s \to \infty$, since problem (QP) is assumed to be bounded below over its feasible region, an optimal solution of the problem exists, so does the corresponding equilibrium point. Hence, for the system defined by (18), we can linearize the system around the neighborhood of an equilibrium point to derive its stability conditions.

Based on Theorems 1 and 2, we now derive a formula of the time-independent delay margin for solving convex quadratic programming problems using the proposed neural network. We proceed as follows. Let $\bar{x}$ be an optimal solution of (QP). Since $A\bar{x} - b = 0$, by linearizing the nonlinear term $\phi(Ax(t-\tau) - b)$ of (19) around the neighborhood of $x(t-\tau) = \bar{x}$, we have the following linear dynamic system: 1

$$\dot{x}(t) \approx -\eta\{Qx(t-\tau) + c + sA^T\phi'(0)(Ax(t-\tau) - b)\} \tag{20}$$

$$= -\eta\{Qx(t-\tau) + c + s\phi'(0)A^T(Ax(t-\tau) - b)\} \tag{21}$$

$$= -\eta(c - s\phi'(0)A^T b) - \eta(Q + s\phi'(0)A^T A)x(t-\tau) \tag{22}$$

where $\phi'(y) = \frac{d\phi(y)}{dy}$. Then, by letting $\tilde{x}$ be an equilibrium point of the system, we have

$$\dot{\tilde{x}} = -\eta(c - s\phi'(0)A^T b) - \eta(Q + s\phi'(0)A^T A)\tilde{x} = 0. \tag{23}$$

Subtracting (23) from (22) and letting $y(t) = x(t) - \tilde{x}$, we have

$$\dot{y}(t) = -\eta(Q + s\phi'(0)A^T A)y(t-\tau) \tag{24}$$

for $t, \tau \geq 0, y(t_0) = \Theta(t_0) - \tilde{x}, t_0 \in [-\tau, 0)$, and $y(0) = \Theta_0 - \tilde{x}$. In $y(t) = x(t) - \tilde{x}$, since $\tilde{x}$ is a constant vector, the stability of (24) implies that of (22), and hence implies the stability of (18).

By comparing (24) with (7), we have $M = 0$ and

$$M_\tau = -\eta(Q + s\phi'(0)A^T A). \tag{25}$$

Since ϕ is a monotone increasing function, we have $\phi'(0) > 0$ and $s\phi'(0) > 0$. Moreover, because both matrices Q and $A^T A$ are symmetric and positive semidefinite, so is matrix $(Q + s\phi'(0)A^T A)$. Consequently, matrix M_τ is symmetric and negative semidefinite. Therefore, matrix M_τ has real nonpositive eigenvalues. Furthermore, by the definition of l_1 in (10), we have

$$l_1 = \mu_p(M) + \|M_\tau\|_p = \|M_\tau\|_p \geq 0 \tag{26}$$

with $p = 1, 2$, and ∞. Therefore, the delay-independent stability condition of Theorem 1 is not satisfied. The stability of an equilibrium point of the system depends on the time delay in the system. As a result of (11), we have

$$l_2 = \mu_p(-\iota M) + \|M_\tau\|_p = \|M_\tau\|_p. \tag{27}$$

Moreover, by Theorem 2, the delay margin can be computed from the following stability condition:

$$\mathrm{Re}\,\lambda_i(M_\tau e^{-\tau\hat{\lambda}}) < 0 \tag{28}$$

for $i = 1, 2, \cdots$, where M_τ is defined in (25) and $\hat{\lambda}$ takes the values of $\iota\omega$, $\|M_\tau\|_p + \iota\omega$, and $r + \iota\|M_\tau\|_p$, with $0 \leq \omega \leq \|M_\tau\|_p$ and $0 \leq r \leq \|M_\tau\|_p$. In particular, if matrix M_τ is diagonalizable, we can solve the delay margin of the linearized system defined by (24) explicitly.

Suppose that matrix M_τ is diagonalizable. Let P and Λ be the matrices of eigenvectors and eigenvalues corresponding to M_τ, respectively, i.e., $M_\tau P = P\Lambda$, where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$ is a diagonal matrix of eigenvalues of M_τ, $\lambda_i \leq 0$, $1 \leq i \leq n$, and P is invertible. Replacing M_τ with $P\Lambda P^{-1}$ in (24), we have

$$\dot{y}(t) = P\Lambda P^{-1} y(t-\tau). \tag{29}$$

Multiplying both sides of the equation by P^{-1}, we have

$$P^{-1}\dot{y}(t) = \Lambda P^{-1} y(t-\tau). \tag{30}$$

By letting $z(t) = P^{-1}y(t)$ and $z(t-\tau) = P^{-1}y(t-\tau)$, we have

$$\dot{z}(t) = \Lambda z(t-\tau). \tag{31}$$

This is equivalent to

$$\dot{z}_i(t) = \lambda_i z_i(t-\tau) \tag{32}$$

for $i = 1, 2, \cdots, n$. Comparing (32) with (13), for $i = 1, 2, \cdots, n$, we have

$$a = 0 \quad \text{and} \quad b = \lambda_i \leq 0. \tag{33}$$

Thus, by (10) and (11), for $i = 1, 2, \cdots, n$

$$l_1 = \mu(a) + |b| = -\lambda_i \geq 0 \quad \text{and} \quad l_2 = \mu(-\iota a) + |b| = -\lambda_i \geq 0. \tag{34}$$

Now, since $a = 0$, by Fig. 1, the equilibrium point of the system defined by (32) is asymptotically stable, if

$$-\frac{\pi}{2\tau} \leq \lambda_i < 0. \tag{35}$$

Consequently, the range of time delay is

$$0 \leq \tau \leq -\frac{\pi}{2\lambda_i} \tag{36}$$

for $i = 1, 2, \cdots, n$ and $\lambda_i < 0$. Therefore, when M_τ is diagonalizable, the delay margin τ_m can be determined by

$$\tau_m = \min_{\substack{1 \leq i \leq n \\ \lambda_i < 0}} -\frac{\pi}{2\lambda_i}. \tag{37}$$

Even if matrix M_τ is not diagonalizable, we can still compute the time delay margin based on the eigenvalues of M_τ. Let $\bar{P}$ and $\bar{\Lambda}$ be the matrices of eigenvectors and eigenvalues of M_τ, respectively, i.e., $M_\tau \bar{P} = \bar{P}\bar{\Lambda}$ and $\bar{P}$ is nonsingular. Then, by the method of generalized eigenvectors, we can always find a matrix $\bar{\Lambda}$ in the Jordan canonical form, i.e.,

$$\bar{\Lambda} = \begin{bmatrix} \Lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \Lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \Lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \Lambda_k \end{bmatrix}$$
$$\text{where } \Lambda_j = \begin{bmatrix} \lambda_j & 1 & 0 & \cdots & 0 \\ 0 & \lambda_j & 1 & \cdots & 0 \\ 0 & 0 & \lambda_j & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_j \end{bmatrix} \tag{38}$$

$\Lambda_j \in R^{m_j \times m_j}$, and $m_j > 0$ for $j = 1, 2, \ldots, k$, and $m_1 + m_2 + \cdots + m_k = n$. By following the same steps as we did for the system with diagonalizable matrix, we have the following system of differential equations for the linearized system defined by (24):

$$\dot{z}(t) = \bar{\Lambda} z(t - \tau) \tag{39}$$

where $\bar{\Lambda}$ is defined by (38). By Theorem 2, we know that the equilibrium point of the above system is asymptotically stable, if

$$\operatorname{Re}\hat{\lambda} = \operatorname{Re}\lambda_i(\bar{\Lambda}e^{-\tau\hat{\lambda}}) < 0 \tag{40}$$

for $i = 1, 2, \cdots$, where $\hat{\lambda}$ takes the values of $\iota\omega$, $||\bar{\Lambda}||_p + \iota\omega$, and $r + \iota||\bar{\Lambda}||_p$, with $0 \leq \omega \leq ||\bar{\Lambda}||_p$ and $0 \leq r \leq ||\bar{\Lambda}||_p$.

By taking advantage of the special structure of $\bar{\Lambda}$ in (38) and letting $z_j(t) = [z_j^1(t) \cdots z_j^{m_j}(t)]^T$, $1 \leq j \leq k$, and $z(t) = [z_1^T(t) \cdots z_k^T(t)]^T$, the system defined by (39) becomes a combination of the following k independent dynamic systems:

$$\dot{z}_j(t) = \Lambda_j z_j(t - \tau) \tag{41}$$

for $j = 1, 2, \cdots, k$. Thus, the delay-dependent stability condition (40) of the system defined by (39) can be simplified to

$$\operatorname{Re}\hat{\lambda} = \operatorname{Re}\lambda_i(\Lambda_j e^{-\tau\hat{\lambda}}) < 0 \tag{42}$$

for $i = 1, 2, \cdots$, and $j = 1, 2, \cdots, k$, where $\hat{\lambda}$ takes the values of $\iota\omega$, $||\Lambda_j||_p + \iota\omega$, and $r + \iota||\Lambda_j||_p$, with $0 \leq \omega \leq ||\Lambda_j||_p$ and $0 \leq r \leq ||\Lambda_j||_p$. Furthermore, by the result of [3, pp. 93–94] because

$$\lambda_i(\Lambda_j e^{-\tau\hat{\lambda}}) = \lambda_i(\lambda_j e^{-\tau\hat{\lambda}}) \tag{43}$$

for $i = 1, 2, \cdots$, and $j = 1, 2, \cdots, k$, the delay-dependent stability condition (42) can be further simplified to

$$\operatorname{Re}\hat{\lambda} = \operatorname{Re}\lambda_i(\Lambda_j e^{-\tau\hat{\lambda}}) = \operatorname{Re}\lambda_i(\lambda_j e^{-\tau\hat{\lambda}}) < 0 \tag{44}$$

where $\hat{\lambda}$ takes the values described by condition (42). Note that, for $1 \leq j \leq k$, the condition $\operatorname{Re}\lambda_i(\lambda_j e^{-\tau\hat{\lambda}}) < 0$ in (44) is very similar to the delay-dependent stability condition of the following dynamic system:

$$\dot{u}(t) = \lambda_j u(t - \tau) \tag{45}$$

except that in the delay-dependent stability condition of the system

$$\operatorname{Re}\hat{\lambda}_u = \operatorname{Re}\lambda_i(\lambda_j e^{-\tau\hat{\lambda}_u}) < 0 \tag{46}$$

where $\hat{\lambda}_u$ takes the values of $\iota\omega$, $|\lambda_j| + \iota\omega$, and $r + \iota|\lambda_j|$, with $0 \leq \omega \leq |\lambda_j|$ and $0 \leq r \leq |\lambda_j|$. Therefore, the stability of an equilibrium point of the system defined by (45) implies that $\operatorname{Re}\lambda_i(\lambda_j e^{-\tau\hat{\lambda}_u}) < 0$ in (46). This further implies the stability of an equilibrium point of the system defined by (41). Consequently, it implies the stability of the system defined by (39). Following the delay-dependent stability condition of Theorem 2, every equilibrium point of the system defined by (39) with a nondiagonalizable matrix M_τ is asymptotically stable, if

$$-\frac{\pi}{2\tau} \leq \lambda_j < 0 \tag{47}$$

for $j = 1, 2, \cdots, k$. Therefore, the range of time delay is

$$0 < \tau \leq -\frac{\pi}{2\lambda_j} \tag{48}$$

for $j = 1, 2, \cdots, k$ and $\lambda_j < 0$, and the delay margin τ_m can be computed as

$$\tau_m = \min_{\substack{1 \leq j \leq k \\ \lambda_j < 0}} -\frac{\pi}{2\lambda_j}. \tag{49}$$

Note that the delay margin defined by (37) and (49) is based on the linearized system defined by (24). For the nonlinear dynamic system defined by (18), due to its nonlinearity and stability, the delay margin of the system may not be equal to the suggested

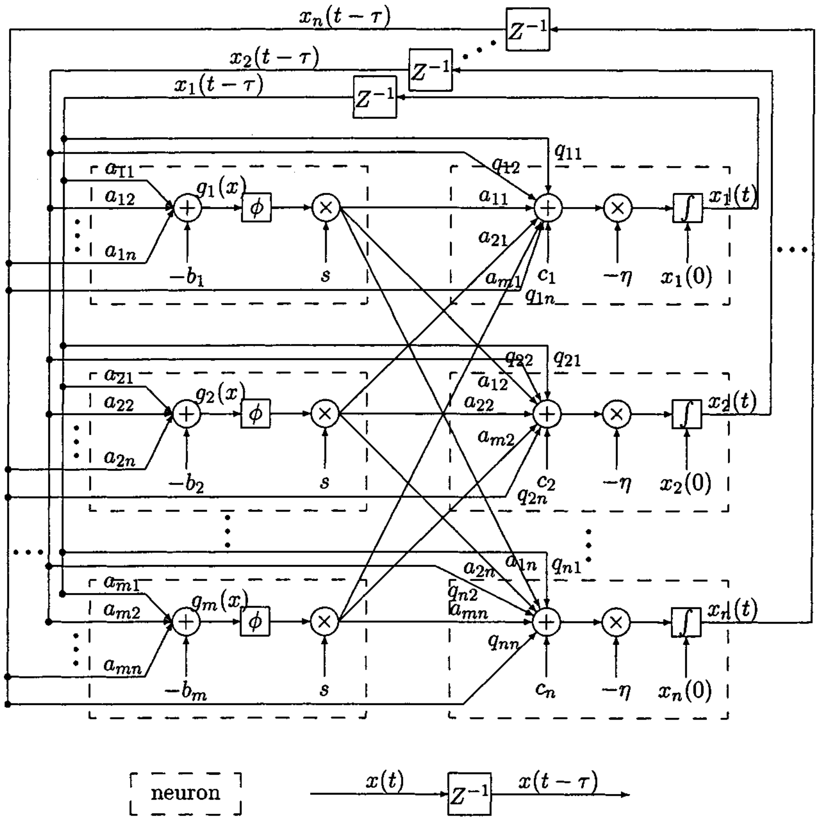


Fig. 2. Neural-network structure for QP problems.

delay margin for initial states outside the neighborhood of its equilibrium point.

IV. Neural-Network Configuration

In order to configurate a neural network for solving convex quadratic programming problems, we have to define its connection weights and neuron activation functions.

Using (18) and (19), we have

$$\begin{aligned}\dot{x}(t) &= -\eta\nabla_{x(t)}E(x(t-\tau),s)\\ &= -\eta\{Qx(t-\tau)+c+sA^T\phi(Ax(t-\tau)-b)\}. \quad (50)\end{aligned}$$

Similar to the McCulloch–Pitts neuron model [15], we use two types of neurons. The first type has a neuron potential $x(t-\tau)$ with coefficients of $x(t-\tau)$ in $Ax(t-\tau)$ as connection weights, b as thresholds, and the derivative of the penalty function $\Phi(\cdot)$ as neuron activation functions. The output of each neuron is computed as the derivative of the penalty function evaluated at the result of $Ax(t-\tau)-b$. The second type neurons take the outputs of the first type and the neuron potential $x(t-\tau)$ as inputs. Such neurons have coefficients of $\phi(Ax(t-\tau)-b)$ in $sA^T\phi(Ax(t-\tau)-b)$ and coefficients of $x(t-\tau)$ in $Qx(t-\tau)$ as connection weights, c as thresholds, and integrators as neuron activation functions after resulting outputs being multiplied by $-\eta$. The outputs of the second type neurons are fed back to the first type as inputs. Then a near-optimal solution to the convex quadratic programming problem can be obtained as an equilibrium point of the corresponding neural dynamics by measuring the output voltage of the proposed neural network with a sufficiently large penalty parameter s. Fig. 2 shows the architecture of the proposed neural network, where $A=[a_{ij}]_{m\times n}$ and $Q=[q_{ij}]_{n\times n}$.

For a convex quadratic programming problem with inequality constraints, i.e.,

$$\begin{aligned}&(\text{QP}\leq)\ \min\frac{1}{2}x^TQx+c^Tx\\ &\text{s.t.}\quad Ax-b\leq 0,\quad x\geq 0\end{aligned}\qquad (51)$$

where $c\in R^n$, $A\in R^{m\times n}$, $b\in R^m$, $x\in R^n$, and $Q\in R^{n\times n}$ is symmetric and positive semidefinite, we can transform the inequality constraints into equality constraints by adding slack variables, i.e.,

$$Ax-b+x_s=0 \qquad (52)$$

where $x_s\in R^m$ and $x_s\geq 0$.

It should be noted that the simple box (bounds) constraints $x_j^{\min}\leq x_j\leq x_j^{\max}$ can be fulfilled by employing limiting integrators with nonlinear (hardware) limiters at their outputs [5]. This means that the input signals of an integrator are integrated but cannot drive the output x_j beyond some specified limits. In such an approach, all box constraints are "hard," i.e., the constraints must not be violated during the whole optimization process.

An alternative is to introduce an unbounded variable u_j, for $j=1,2,\cdots,n$, that provides a nonlinear transformation

$$x_j=h_j(u_j). \qquad (53)$$

For example, for a given $\gamma > 0$, we may choose

$$x_j = x_j^{\min} + \frac{x_j^{\max} - x_j^{\min}}{1 + e^{-\gamma u_j}}. \tag{54}$$

V. Numerical Examples

To demonstrate the behavior and properties of the proposed neural network, we conduct experiments on four constrained quadratic programming problems. All four problems share the following set of constraints:

$$5x_1 + 3x_2 \le 30, \quad 2x_1 - 3x_2 \ge -9, \quad x_1 \le 5, \quad x_2 \le 5, \quad \text{and } x_1 \ge 0, x_2 \ge 0. \tag{55}$$

Let F represent a set of feasible solutions defined by these constraints. It is not difficult to see that F has five extreme points, (0,0), (5,0), (5,5/3), (3,5), and (0,3).

Let $x = (x_1, x_2)^T$ be a column vector. The objective of the first problem (P_1) is to minimize a a degenerate convex quadratic (linear) function

$$f_1(x) = -x_1 - x_2 \tag{56}$$

subject to $x \in F$ with Q being a zero matrix in (1). It can be seen that the problem has a unique optimal solution at the extreme point (3,5).

The objective of the second quadratic programming problem (P_2) is to minimize

$$f_2(x) = \frac{1}{2}(x_1 - 2)^2 + \frac{1}{2}(x_2 - 2)^2 \tag{57}$$

subject to $x \in F$. Because an unconstrained $f_2(\cdot)$ attains its minimum at $(2, 2) \in F$, problem (P_2) has an optimal solution (2, 2) lying in the interior of F.

The third quadratic programming problem (P_3) has an objective function

$$f_3(x) = \frac{1}{2}\left(x_1 - \frac{5}{6}\right)^2 + \frac{1}{2}(x_2 - 5)^2 \tag{58}$$

subject to $x \in F$. Because an unconstrained $f_3(\cdot)$ attains its minimum at $(5/6, 5) \notin F$, problem (P_3) has an optimal solution (3/2,4) sitting on the boundary of F.

The forth quadratic programming problem (P_4) has an objective function

$$f_4(x) = 10(x_2 - 2x_1)^2 + (1 - x_1)^2 \tag{59}$$

subject to $x \in F$ with its minimum at (1,2). Function $f_4(\cdot)$ takes the convex form of the following nonlinear banana function (also called Rosenbrock's function) [11]:

$$f(x) = 100\left(x_2 - 2x_1^2\right)^2 + (1 - x_1)^2 \tag{60}$$

by reducing the order of x_1 in $(x_2 - 2x_1^2)^2$ by one. This function is notorious in optimization because it causes slow convergence for most solution methods.

To apply the neural network technology to solve these problems, the inequality constraints are transformed into equality constraints by introducing nonnegative slack variables x_3, x_4, x_5, and x_6, i.e., let

$$\begin{aligned} g_1(x) &= 5x_1 + 3x_2 + x_3 - 30 = 0 \\ g_2(x) &= 2x_1 - 3x_2 - x_4 + 9 = 0 \\ g_3(x) &= x_1 + x_5 - 5 = 0 \\ g_4(x) &= x_2 + x_6 - 5 = 0 \\ & \qquad x_1, x_2, \cdots, x_6 \ge 0. \end{aligned} \tag{61}$$

Then, the state variables of the proposed neural network become $x(t) = [x_1(t), x_2(t), \cdots, x_6(t)]^T$. Our numerical experiments start with the initial state (point) at $x^0 = [0, 0, 0, 0, 0, 0]^T$.

Let $\eta = 1, s = 10$

$$\Phi(g_i(x(t))) = 10 \ln \cosh \frac{g_i(x(t))}{10} \tag{62}$$

and

$$\phi(g_i(x(t))) = \frac{\partial \Phi(g_i(x(t)))}{\partial g_i(x(t))} = \tanh \frac{g_i(x(t))}{10} \tag{63}$$

with $g_i(x(t)) = A_i x(t) - b_i$, for $i = 1, 2, 3$, and 4. The following energy function is formulated for solving problem (P_1) with $t \ge 0$ in (64) shown at the bottom of the page. Taking derivative of the energy function with respect to $x(t)$ and replacing $x(t)$ in $E(x(t), s)$ by $x(t - \tau)$, we have the following

$$\begin{aligned} E(x(t), s) &= f_1(x(t)) + s\sum_{i=1}^{4} \Phi(g_i(x(t))) \\ &= f_1(x(t)) + s\sum_{i=1}^{4} 10 \ln \cosh \frac{g_i(x(t))}{10} \\ &= -x_1(t) - x_2(t) + 100\{\ln\cosh[0.1(5x_1(t) + 3x_2(t) + x_3(t) - 30)] \\ &\quad + \ln\cosh[0.1(2x_1(t) - 3x_2(t) - x_4(t) + 9)] \\ &\quad + \ln\cosh[0.1(x_1(t) + x_5(t) - 5)] \\ &\quad + \ln\cosh[0.1(x_2(t) + x_6(t) - 5)]\}. \end{aligned} \tag{64}$$

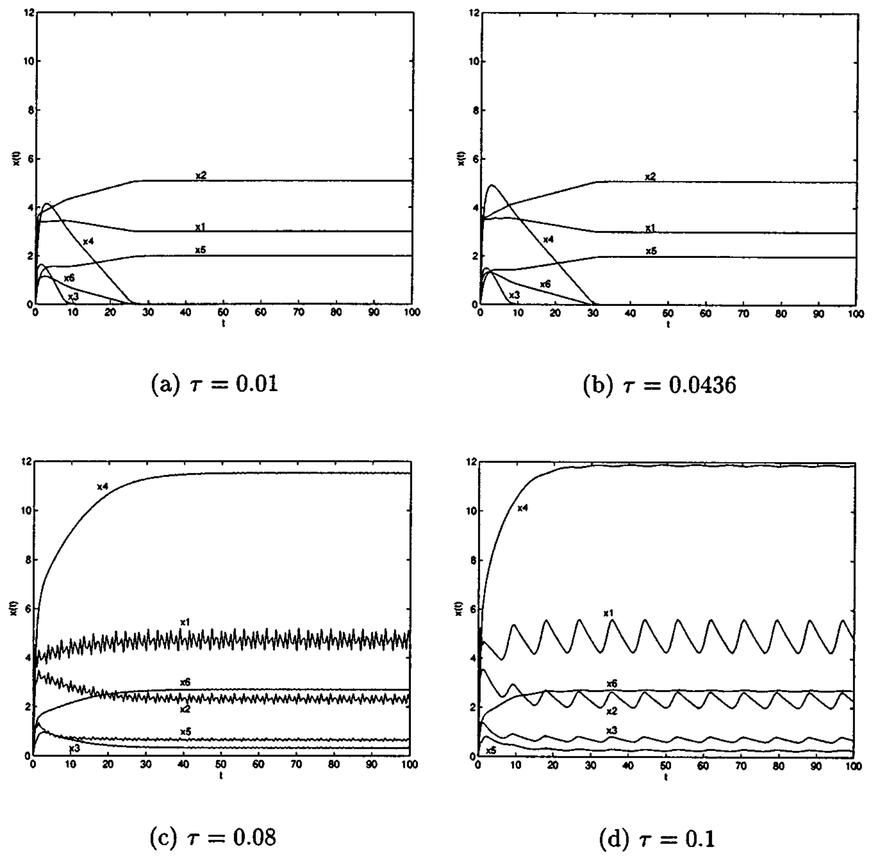


Fig. 3. State trajectories of problem (P_1) for different time delays with $\Theta(t_0) = x^4$, $t_0 \in [-\tau, 0]$, and $t \in [0, 100]$.

system of difference-differential equations for the neural dynamics shown in (65) at the bottom of the page. Consequently

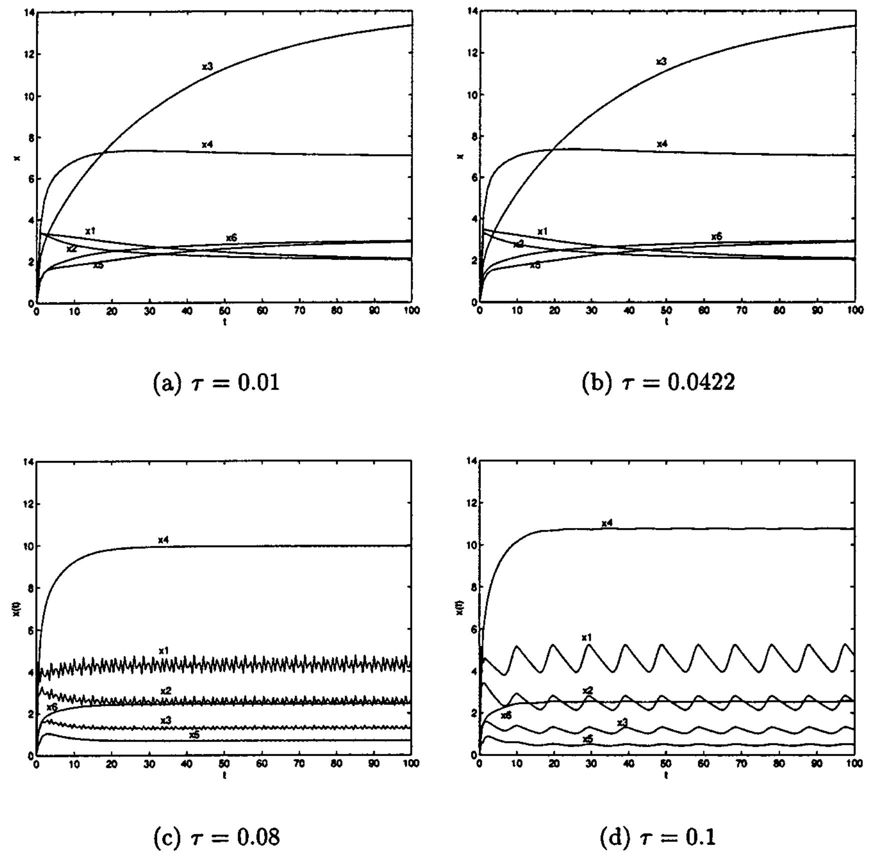
$$\dot{x}_1(t) = 1 - 10 \left(5 \tanh \frac{g_1(x(t-\tau))}{10} + 2 \tanh \frac{g_2(x(t-\tau))}{10} + \tanh \frac{g_3(x(t-\tau))}{10}\right) \tag{66}$$

with

$$\dot{x}_2(t) = 1 - 10 \left(3 \tanh \frac{g_1(x(t-\tau))}{10} - 3 \tanh \frac{g_3(x(t-\tau))}{10} + \tanh \frac{g_4(x(t-\tau))}{10}\right) \tag{67}$$

$$\dot{x}_3(t) = -10 \tanh \frac{g_1(x(t-\tau))}{10} \tag{68}$$

$$\begin{aligned}
\dot{x}(t) &= -\eta \nabla_{x(t)} E(x(t-\tau), s) \\
&= -\eta \left\{ \nabla_{x(t)} f_1(x(t-\tau)) + s \sum_{i=1}^{4} \phi(g_i(x(t-\tau))) \nabla x_i(t) g_i(x(t-\tau)) \right\} \\
&= -\eta \nabla_{x(t)} f_1(x(t-\tau)) - \eta s \sum_{i=1}^{4} \phi(g_i(x(t-\tau))) \nabla x_i(t) g_i(x(t-\tau)) \\
&= \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - 10 \begin{bmatrix} 5 \tanh \frac{g_1(x(t-\tau))}{10} + 2 \tanh \frac{g_2(x(t-\tau))}{10} + \tanh \frac{g_3(x(t-\tau))}{10} \\ 3 \tanh \frac{g_1(x(t-\tau))}{10} - 3 \tanh \frac{g_3(x(t-\tau))}{10} + \tanh \frac{g_4(x(t-\tau))}{10} \\ \tanh \frac{g_1(x(t-\tau))}{10} \\ - \tanh \frac{g_2(x(t-\tau))}{10} \\ \tanh \frac{g_3(x(t-\tau))}{10} \\ \tanh \frac{g_4(x(t-\tau))}{10} \end{bmatrix}.
\end{aligned} \tag{65}$$



(a) $\tau = 0.01$ (b) $\tau = 0.0422$

(c) $\tau = 0.08$ (d) $\tau = 0.1$

Fig. 4. State trajectories of problem (P_2) for different time delays with $\Theta(t_0) = x^0$, $t_0 \in [-\tau, 0]$, and $t \in [0, 100]$.

and

$$\dot{x}_4(t) = 10 \tanh \frac{g_2(x(t-\tau))}{10} \tag{69}$$

$$\dot{x}_5(t) = -10 \tanh \frac{g_3(x(t-\tau))}{10} \tag{70}$$

$$\dot{x}_6(t) = -10 \tanh \frac{g_4(x(t-\tau))}{10}. \tag{71}$$

To determine the time delay margin of the neural dynamics for problem (P_1) using (25), we have

$$M_\tau^1 = -\eta s \phi'(0) A^T A \tag{72}$$

where $\phi'(g_i(x(t))) = 1/10 \,\mathrm{sech}(g_i(x(t))/10)$. Therefore, when $\eta = 1$, $s = 10$, and $g_i(x(t)) = 0$, $1 \le i \le m$, we have $\phi'(0) = 1/10$, and

$$M_\tau^1 = -A^T A = \begin{bmatrix} -30 & -9 & -5 & 2 & -1 & 0 \\ -9 & -19 & -3 & -3 & 0 & -1 \\ -5 & -3 & -1 & 0 & 0 & 0 \\ 2 & -3 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{73}$$

Since the rank of matrix M_τ^1 is four with eigenvalues approximately being –1, –1, 0, 0, –14.95, and –36.05, therefore M_τ^1 is not diagonalizable. By (49), the time delay margin τ_m^1 of (P_1) is given by

$$\tau_m^1 = \min_{\substack{1 \le j \le k \\ \lambda_j < 0}} -\frac{\pi}{2\lambda_j} \approx \min\{1.5708, 0.1051, 0.0436\} = 0.0436. \tag{74}$$

We use the Fortran program developed by Lo and Jackiewicz [22] in 1992 to simulate the neural dynamics of the proposed neural network with time delay. The program is compiled and executed on a SUN Sparc ES4000 computer. The program uses variable-step and variable-order Adams methods to solve a system of neutral delay differential equations with state-dependent delays (SNDDE's).

Fig. 3 shows the stability of solutions to problem (P_1) for different values of the time delay τ, using the computer program. The initial value function in the simulation is $\Theta(t_0) = x^0$, $t_0 \in [-\tau, 0]$, where τ is the time delay of each plot in the figure. Simulation time is from $t = 0$ to 100. The figure gives four plots of the state trajectories for time delay being 0.01,0.0436, 0.08 and 0.1, respectively. From our formula, the neural states start to become unstable, when $\tau > 0.0436$. Fig.(c) and (d) clearly support this fact.

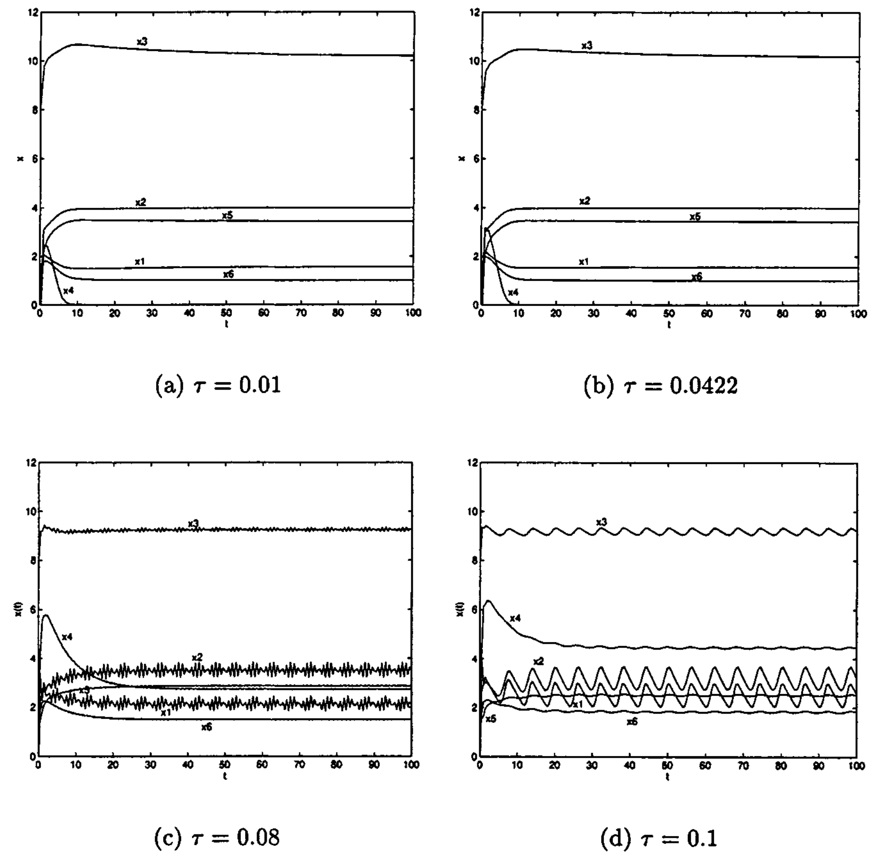


Fig. 5. State trajectories of problem (P_3) for different time delays with $\Theta(t_0) = x^0$, $t_0 \in [-\tau, 0]$, and $t \in [0, 100]$.

For (P_2) and (P_3), because

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

by (25), we have

$$M_\tau^2 = M_\tau^3 = -\eta(Q + s\phi'(0)A^T A) = \begin{bmatrix} -31 & -9 & -5 & 2 & -1 & 0 \\ -9 & -20 & -3 & -3 & 0 & -1 \\ -5 & -3 & -1 & 0 & 0 & 0 \\ 2 & -3 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{75}$$

Since matrices M_τ^2 and M_τ^3 have full rank with eigenvalues being approximately –1, –1, –0.0296, –0.1091, –16.5993, and –37.2619, these matrices are diagonalizable. Hence, by (37), the time delay margins τ_m^2 of (P_2) and τ_m^3 of (P_3) are given by

$$\tau_m^2 = \tau_m^3 \approx \min\{1.5708, 53.0135, 14.3939, 0.0946, 0.0422\} = 0.0422. \tag{76}$$

Figs. 4 and 5 depict the neural state trajectories of problems (P_2) and (P_3), respectively. The initial value function used here is the same as that of problem (P_1). The simulation time ranges from $t = 0$ to 100. Each of these figures includes four plots of the state trajectories with time delay of 0.01, 0.0422, 0.08, and 0.1, respectively. It is clearly seen that the neural states become unstable, when $\tau > 0.0422$.

Finally, for problems (P_4), we have

$$Q = \begin{bmatrix} 82 & -40 & 0 & 0 & 0 & 0 \\ -40 & 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

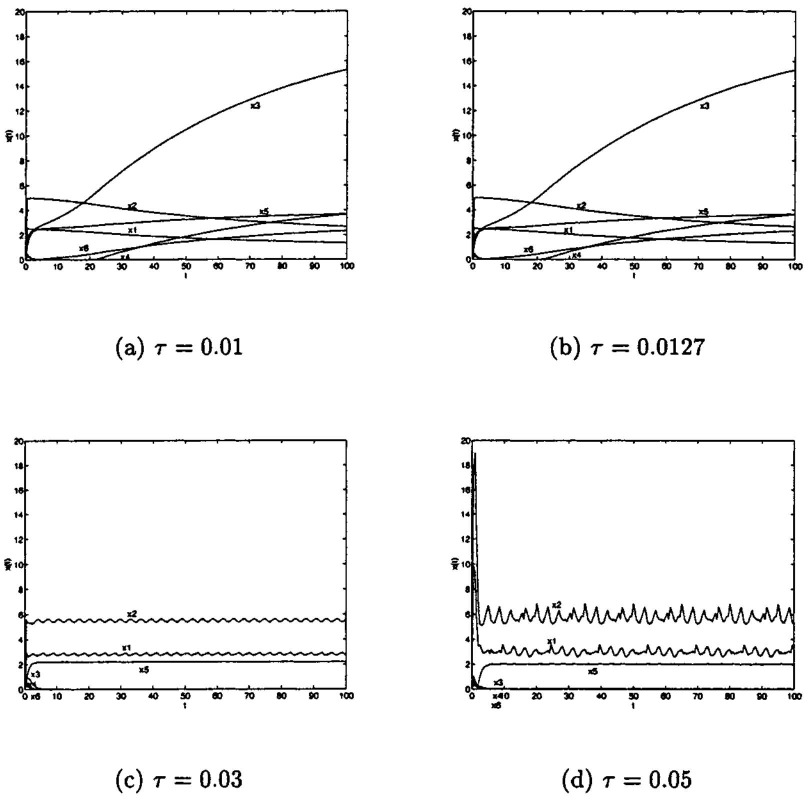
and

$$M_\tau^4 = -(Q + A^T A) = \begin{bmatrix} -112 & 31 & -5 & 2 & -1 & 0 \\ 31 & -39 & -3 & -3 & 0 & -1 \\ -5 & -3 & -1 & 0 & 0 & 0 \\ 2 & -3 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{77}$$

Since M_τ^4 has full rank with eigenvalues being approximately –1, –1, –0.8533, –0.0133, –28.5589 and –123.5745, M_τ^4 is diagonalizable. Using (37), we have the time-delay margin

$$\tau_m^4 \approx \min\{1.5708, 1.8409, 118.2573, 0.0550, 0.0127\} = 0.0127. \tag{78}$$

Fig. 6 depicts the neural state trajectories for this problem with the time delay being 0.01, 0.0127, 0.03, and 0.05. It is clearly seen that the states become unstable, when $\tau > 0.0127$. Also notice that the slow convergence around the optimal solution of



(a) $\tau = 0.01$ (b) $\tau = 0.0127$

(c) $\tau = 0.03$ (d) $\tau = 0.05$

Fig. 6. State trajectories of problem (P_4) for different time delays with $\Theta(t_0) = x^0$, $t_0 \in [-\tau, 0]$, and $t \in [0, 100]$.

the problem requires a smaller time delay margin than the delay margin in the previous two cases, i.e., $0.0127 < 0.0422$.

Putting things together, we observed that, if the time delay is less than or equal to the proposed delay margin computed by (37) and (49), the equilibrium point of the neural dynamics defined by (18) is stable. If the time delay is greater than the delay margin, as shown in Figs. 3–6, the states of the neural networks become unstable. Moreover, the states become completely unstable, when the time delay is much greater than the delay margin. Therefore, the proposed delay margin given by (37) and (49) indeed provides good estimates of the delay margin for our experiments.

VI. Conclusion

In the paper, a neural-network computational scheme with time delay has been proposed for solving convex quadratic programming problems. A delay margin for the stability of an equilibrium is derived. The configuration and some operational characteristics of the proposed neural network are demonstrated via numerical examples. The findings are positively supported by computer simulation results.

References

[1] P. Baldi and A. F. Atiya, "How delays affect neural dynamics and learning," *IEEE Trans. Neural Networks*, vol. 5, no. 4, pp. 612–621, 1994.

[2] R. Bellman and K. L. Cooke, *Differential-Difference Equations*. New York: Academic, 1963.

[3] Y.-H. Chen, "Neural network technology for solving convex programming problems and its application," Ph.D. dissertation, North Carolina State Univ., Raleigh, NC, 1997.

[4] Y.-H. Chen and S.-F. Fang, "Solving convex programming problems with equality constraints by neural networks," *Comput. Math. Applicat.*, vol. 36, no. 7, pp. 41–68, 1998.

[5] A. Cichocki, R. Unbehauen, K. Weinzierl, and R. Hölzel, "A new neural network for solving linear programming problems," *European J. Operational Res.*, vol. 93, pp. 244–256, 1996.

[6] K. L. Cooke and Z. Grossman, "Discrete delay, distributed delay, and stability switches," *J. Math. Anal. Applicat.*, vol. 86, pp. 592–627, 1982.

[7] J. P. Coughlin and R. H. Baran, *Neural Computation in Hopfield Networks and Boltzmann Machines*. Newark, DE: Univ. Delaware Press, 1995.

[8] R. Datko, "Some second-order vibrating systems cannot tolerate small time delays in their damping," in *Proc. 28th Conf. Decision Contr.*, Tampa, FL, 1969, pp. 2032–2033.

[9] J. B. Dennis, *Mathematical Programming and Electrical Networks*. London, U.K.: Chapman and Hall, 1959.

[10] S.-C. Fang and S. C. Puthenpura, *Linear Optimization and Extensions: Theory and Algorithms*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[11] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, *Computation Methods for Mathematical Computations*. Englewood Cliffs, NJ: Prentice-Hall, 1977.

[12] K. Gopalsamy and X.-Z. He, "Stability in asymmetric Hopfield nets with transmission delays," *Phys. D*, vol. 76, pp. 344–358, 1994.

[13] K. Gopalsamy and X.-Z. He, "Delay-independent stability in bidirectional associative memory networks," *IEEE Trans. Neural Networks*, vol. 5, no. 6, pp. 998–1002, 1994.

[14] J. K. Hale, E. F. Infante, and F.-S. Tsen, "Stability in linear delay equations," *J. Math. Anal. Applicat.*, vol. 105, pp. 533–555, 1985.

[15] S. Haykin, *Neural Networks, A Comprehensive Foundation*. New York: Macmillan, 1994.

[16] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Academy Sci. USA.*, vol. 79, 1982, pp. 2554–2558.

[17] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," in *Proc. Nat. Academy Sci. USA.*, vol. 81, 1984, pp. 3088–3092.

[18] J. J. Hopfield and D. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 58, pp. 63–70, 1985.

[19] K. Ito, H. T. Tran, and A. Manitius, "A fully-discrete special method for delay-differential equations," *SIAM J. Numer. Anal.*, vol. 28, no. 4, pp. 1121–1140, 1991.

[20] E. W. Kamen, "On the relationship between zero criteria for two-variable polynomials and asymptotic stability of delay differential equations," *Int. J. Contr.*, vol. AC-25, no. 5, pp. 983–984, 1980.

[21] J. H. Lee, W. H. Kwon, and J. W. Lee, "Delay independent stability conditions for variable delay systems and its application to delay margins," in *32nd IEEE Conf. Decision Contr.*, 1997.

[22] E. Lo and Z. Jackwicz, "SNDDELM algorithm for the numerical solution of systems of neutral delay differential equations with state-dependent delays (SNDDE) by adams predictor-corrector methods," *FORTRAN Program*, 1992.

[23] C.-Y. Maa and M. A. Shanblatt, "Linear and quadratic programming neural network analysis," *IEEE Trans. Neural Networks*, vol. 3, no. 4, pp. 580–594, 1992.

[24] C.-Y. Maa and M. A. Shanblatt, "A two-phase optimization neural network," *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 1003–1010, 1992.

[25] C. M. Marcus and R. M. Westervelt, "Stability of analog neural networks with delay," *Phys. Rev. A*, vol. 39, no. 1, pp. 347–359, 1989.

[26] C. M. Marcus and R. M. Westervelt, "Stability and convergence of analog neural networks with multiple-time-step parallel dynamics," *Phys. Rev. A*, vol. 42, no. 4, pp. 2410–2417, 1990.

[27] C. M. Marcus, F. R. Waugh, and R. M. Westervelt, "Nonlinear dynamics and stability of analog neural networks," *Phys. D*, vol. 51, no. 1, pp. 234–247, 1991.

[28] T. Mori, "Criteria for asymptotic stability of linear time-delay systems," *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 2, pp. 158–161, 1985.

[29] T. Mori, N. Fukuma, and M. Kuwahara, "Simple stability criteria for single and composite linear systems with time delays," *Int. J. Contr.*, vol. 34, no. 6, pp. 1175–1184, 1981.

[30] T. Mori, N. Fukuma, and M. Kuwahara, "On an estimate of the decay rate for stable linear delay systems," *Int. J. Contr.*, vol. 36, no. 1, pp. 95–97, 1982.

[31] T. Mori and H. Kokame, "Stability of $\dot{x}(t) = ax(t) + bx(t - \tau)$," *IEEE Trans. Automat. Contr.*, vol. 34, no. 4, pp. 460–462, 1989.

[32] B. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Trans. Neural Networks*, vol. 6, no. 5, pp. 1212–1228, 1995.

[33] A. J. Pritchard and D. Salamon, "The linear quadratic optimal control problem for infinite dimensional systems with unbounded input and output operators," *SIAM J. Contr. Optimi.*, vol. 25, pp. 121–199, 1987.

[34] T. E. Stern, *Theory of Nonlinear Networks and Systems*. New York: Addison-Wesley, 1965.

[35] A. G. Tsirukis, G. V. Reklaitis, and M. F. Tenorio, "Nonlinear optimization using generalized Hopfield networks," *Neural Comput.*, vol. 1, no. 4, pp. 511–521, 1989.

[36] J. Wang, "Deterministic annealing neural network for convex programming," *Neural Networks*, vol. 7, no. 4, pp. 629–641, 1994.

[37] Z. Yi, "Global exponential stability and periodic solutions of delay Hopfield neural networks," *Int. J. Syst. Sci.*, vol. 27, no. 2, pp. 227–231, 1996.

[38] Z. Yi, S. M. Zhong, and Z. L. Li, "Periodic solutions and stability of Hopfield neural networks with variable delays," *Int. J. Syst. Sci.*, vol. 27, no. 9, pp. 895–901, 1996.

[39] S. H. Zak, V. Upatising, and S. Hui, "Solving linear programming problems with neural networks: A comparative study," *IEEE Trans. Neural Networks*, vol. 6, no. 1, pp. 96–104, 1995.

[40] Y. Zhang and P. Banks, "Stability of nonlinear analytic delay equations," *Int. J. Syst. Sci.*, vol. 25, no. 11, pp. 2015–2022, 1994.