Pergamon

Computers Math. Applic. Vol. 36, No. 7, pp. 41–68, 1998

Printed in Great Britain
0898-1221/98 $19.00 + 0.00

PII: S0898-1221(98)00172-2

Solving Convex Programming Problems with Equality Constraints by Neural Networks

Y.-H. CHEN
Operations Research, Fruit of the Loom
Bowling Green, KY 42102, U.S.A.
ychen@fruit.com

S.-C. FANG
Industrial Engineering and Operations Research
North Carolina State University
Raleigh, NC 27695-7906, U.S.A.
fang@eos.ncsu.edu

(Received February 1998; accepted March 1998)

Abstract—This paper presents a neural network approach for solving convex programming problems with equality constraints. After defining the energy function and neural dynamics of the proposed neural network, we show the existence of an equilibrium point at which the neural dynamics becomes asymptotically stable. It is shown that under proper conditions, an optimal solution of the underlying convex programming problems is an equilibrium point of the neural dynamics, and vise versa. The configuration of the proposed neural network with an exact layout is provided for solving linear programming problems. The operational characteristics of the neural network are demonstrated by numerical examples. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords—Convex programming, Penalty function, Artificial neural networks, Hopfield networks.

1. INTRODUCTION

In real world situations, a large class of logical problems, especially those in engineering and economics, can be formulated as optimization problems. While the majority of these optimization problems are solved off-line, an on-line optimizer is required or desirable for many real-time applications which may have an immense number of variables with combinatorial nature. Most digital computers would be short of power and speed to perform such tasks [1–3].

For computational efficiency, an analog circuit, also known as dynamic solver or analog computer, was first proposed by Dennis [4] in 1959, and further extended by Stern [5] in 1965. Later, in 1982, the *Hopfield network* [2] was proposed to solve a 30-city Traveling Salesman Problem (TSP) [6]. Thereafter, various stochastic and deterministic methods were developed for applying the Hopfield network to solve optimization problems [7–12].

The objective of this paper is to examine the theoretical properties of a proposed neural network for solving convex programming problems with equality constraints. The neural network is a generalization of Kennedy and Chua's model [13] proposed in 1987. Our problem is formulated

This research work was partially supported by the 1996 National Textile Center Research Grant.

in Section 2. After defining the neural energy function and the neural dynamics of the proposed neural network, we show the existence of an equilibrium point at which the neural dynamics becomes asymptotically stable in Section 3. An equivalence relationship between the optimal solutions of the underlying convex programming problems and the equilibrium points of the neural dynamics is also established in this section. Section 4 provides the configuration and numerical examples of the proposed neural network for solving linear programming problems. Conclusions and comments are given in Section 6 to conclude the paper.

2. PROBLEM FORMULATION

Consider the following constrained convex programming problem:

$$\text{(P)} \qquad \begin{aligned} &\min f(x), \\ &\text{s.t.} \quad g(x) = 0, \\ &\quad x \in X, \end{aligned} \tag{1}$$

where X is a subset of R^n, $f : R^n \to R$, and $g : R^n \to R^m$ with $g(x) = [g_i(x)]^\top$, $i = 1, 2, \ldots, m$. Functions f and g_i, $i = 1, 2, \ldots, m$, are convex and continuously differentiable. Assume that the feasible domain

$$F = \{x \in R^n \mid g(x) = 0 \text{ and } x \in X\}$$

is not empty and the objective function $f(x)$ is bounded below over F.

To apply the Hopfield network to solve Problem (P), as proposed by Kennedy and Chua [13] in 1987, we relax a constrained optimization 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

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

and $\Phi(\cdot)$ is convex and piecewise differentiable. Typical examples such as the quadratic, p-norm, Huber's, and logistic penalty functions can be found in [14].

In order to analyze the stability of a neural network corresponding to Problem (P), the energy function of the neural network is required to be continuously differentiable. Since the penalty function will become part of the proposed neural energy function, we consider only those penalty functions which are convex and continuously differentiable over R.

Once the penalty function is determined, Problem (P) can be reformulated as the following constrained convex optimization problem:

$$\text{(P}_\Phi\text{)} \qquad \begin{aligned} &\min f(x), \\ &\text{s.t.} \quad \sum_{i=1}^{m} \Phi\left(g_i(x)\right) = 0, \\ &\quad x \in X. \end{aligned} \tag{4}$$

By incorporating the explicit constraints into the objective function $f(x)$, Problem (P$_\Phi$) can be transformed into the following unconstrained convex programming problem [10,11,13,15–19]:

$$\min_{x \in X} E(x, s), \tag{5}$$

where

$$E(x, s) = f(x) + s \sum_{i=1}^{m} \Phi\left(g_i(x)\right), \tag{6}$$

and $s > 0$ is a penalty parameter. The problem gives an energy function for the Hopfield network corresponding to Problem (P). Note that all decision variables in (P) become state variables in the energy function. They are actually time-dependent [2,3], i.e., $x = x(t)$, $t \geq 0$, and our objective is to find a solution to

$$\min_{x(t) \in X} E(x(t), s), \tag{7}$$

where

$$E(x(t), s) = f(x(t)) + s \sum_{i=1}^{m} \Phi\left(g_i\left(x(t)\right)\right). \tag{8}$$

We then develop a neural network for Problem (P) based on the derived energy function.

By assuming that $E(x(t), s)$ increases unboundedly as $\|x(t)\|_p \to \infty$, Kennedy and Chua showed that $E(x(t), s)$ is a Liapùnov function [16] based on the network analysis of nonlinear circuit theory derived in [14].

3. RESULT ANALYSIS

A general penalty method considers the following nonlinear optimization problem [20,21]:

$$\begin{aligned} &\min f(x), \\ &\text{s.t.} \quad x \in V, \end{aligned} \tag{9}$$

where f is a continuous function on R^n and V is a constraint set in R^n. Function f is assumed to be bounded below over the feasible region V. In most applications V is defined explicitly by a number of functional constraints, but in this section the more general description of V is applicable. The idea of the penalty method is to replace (9) by an unconstrained problem of the form

$$\min_{x \in R^n} f(x) + sp(x), \tag{10}$$

where s is a positive constant and p is a function on R^n such that:

(i) p is continuous,
(ii) $p(x) \geq 0$, for all $x \in R^n$, and
(iii) $p(x) = 0$ if and only if $x \in V$.

The global convergence of the penalty method follows from Luenberger [20].

THEOREM 3.1. *Let $\{x_k\}$ be a sequence generated by the penalty method with $\{s_k\}$, $k = 1, 2, \ldots$, being a nonnegative and strictly increasing sequence tending to infinity. Then, any limit point of the sequence $\{x_k\}$ is a solution to (9).*

Since the proposed neural energy $E(x(t), s)$ defined by (8) fits the unconstrained problem $f(x) + sp(x)$ of the penalty method for a fixed penalty parameter, we have the following results.

THEOREM 3.2. *For Problems (P) and (P_Φ) and the energy function $E(x(t), s)$ defined by (8), let $\{s_k\}_1^\infty$ be a nonnegative and strictly increasing sequence tending to infinity, and $\hat{x}_k$ be the minimizer of $E(x(t), s_k)$. Then, any limit point of the sequence $\{\hat{x}_k\}_1^\infty$ is an optimal solution to (P_Φ) and equivalently to (P).*

The following corollary is a direct consequence of the theorem.

COROLLARY 3.1. *With the same settings as in Theorem 3.2, given $\epsilon > 0$, there exists a sufficiently large s such that the minimizers of $E(x(t), s)$ lie in $N(O, \epsilon)$, where $N(O, \epsilon) = \{x \in R^n \mid \|x - \bar{x}\| < \epsilon$, for some $\bar{x} \in O\}$ and O is the set of minimizers of (P).*

To solve Problem (P), let us define the dynamics of the proposed neural network to be

$$\dot{x}(t) = \frac{dx(t)}{dt} = -\eta \nabla_{x(t)} E(x(t), s), \tag{11}$$

where $t \geq 0$, η is a learning coefficient of the neural network with $\eta > 0$, and

$$\nabla_{x(t)} E(x(t), s) = \nabla_{x(t)} f(x(t)) + s \sum_{i=1}^{m} \nabla_{x(t)} \Phi\left(g_i\left(x(t)\right)\right). \tag{12}$$

By letting $\phi(y) = \frac{d\Phi(y)}{dy}$, since $\nabla_{x(t)} \Phi(g_i(x(t))) = \phi(g_i(x(t))) \nabla_{x(t)} g_i(x(t))$, we have

$$\nabla_{x(t)} E(x(t), s) = \nabla_{x(t)} f(x(t)) + s \sum_{i=1}^{m} \phi\left(g_i\left(x(t)\right)\right) \nabla_{x(t)} g_i(x(t)). \tag{13}$$

Consequently,

$$\dot{x}(t) = -\eta \nabla_{x(t)} f(x(t)) - \eta s \sum_{i=1}^{m} \phi\left(g_i\left(x(t)\right)\right) \nabla_{x(t)} g_i(x(t)). \tag{14}$$

Now, for an autonomous dynamic system [22] described by

$$\dot{x}(t) = h(x(t)), \tag{15}$$

where $t \geq 0$, $x(t) = [x_i(t)]^\top$ is a state vector, and $h = [h_i(x(t))]^\top$, for $i = 1, 2, \ldots, n$, if each h_i is assumed to be a C^1 function, the solution of the system exists and is unique for some given initial conditions [22]. Regarded as a function of t in the n-dimensional state space, the solution $x(t)$ of equation (15) is called a *trajectory* or *motion*.

Let us introduce some essential definitions for further discussions.

DEFINITION 3.1. EQUILIBRIUM. *If a vector $\tilde{x}$ is a solution of the dynamic system (15) for a given $t_0 \geq 0$ with $x(t_0) = x_0$, i.e., a solution to the following equation:*

$$\dot{x}(t) = h(x(t)) = 0, \qquad \text{for } t \geq t_0 \quad \text{and} \quad x(t_0) = x_0, \tag{16}$$

then $\tilde{x}$ is said to be an equilibrium point, critical point, or steady state of the dynamic system.

DEFINITION 3.2. STABILITY. *An equilibrium point $\tilde{x}$ is said to be "stable" or "stable in the sense of Liapùnov", if for any positive scalar ϵ, there exists a positive scalar δ such that $\|x(t_0) - \tilde{x}\| < \delta$ implies $\|x(t) - \tilde{x}\| < \epsilon$ for $t \geq t_0$.*

DEFINITION 3.3. ASYMPTOTIC STABILITY. *An equilibrium point $\tilde{x}$ is said to be "asymptotically stable" or "asymptotically stable in the sense of Liapùnov", if the equilibrium point is stable and $\lim_{t \to \infty} x(t) = \tilde{x}$.*

DEFINITION 3.4. LIAPÙNOV FUNCTION. *A Liapùnov function or energy function is a function $E(x(t))$, which satisfies the following conditions.*

(1) *$E(x(t))$ and each partial derivative $\frac{\partial E(x(t))}{\partial x_i(t)}$, $1 \leq i \leq n$, are continuous.*
(2) *$E(x(t))$ is nonnegative, i.e., $E(x(t)) \geq 0$. Especially, $E(\tilde{x}) = 0$ and $E(x(t)) > 0$ for $x(t)$ in some neighborhood of the equilibrium point $\tilde{x}$, i.e., $\{x \in R^n \mid \|x - \tilde{x}\| \leq \epsilon$, for some small $\epsilon > 0\}$.*
(3) *The derivative of $E(x(t))$ with respect to time t is nonpositive, namely,*

$$\frac{dE(x(t))}{dt} = \left[\nabla_{x(t)} E(x(t))\right]^\top \dot{x}(t) = \left[\nabla_{x(t)} E(x(t))\right]^\top h(x(t))$$

is nonpositive for $x(t) \in \{x \in R^n \mid \|x - \tilde{x}\| \leq \epsilon$, for some small $\epsilon > 0\}$, and $\frac{dE(x(t))}{dt} = 0$ for $x(t) = \tilde{x}$.

These definitions are usually referred to the Russian mathematician Liapùnov for his important work of 1892 [23]. For the stability analysis of (15), we have the following theorems of the Liapùnov theory [22,24].

THEOREM 3.3. *An equilibrium point of equation (15) is stable, if there exists a Liapùnov function associated with the system.*

THEOREM 3.4. *An equilibrium point of equation (15) is asymptotically stable, if there exists a Liapùnov function whose derivative* $\frac{dE(x(t))}{dt}$ *is negative for* $x(t) \neq \tilde{x}$ *and* $x(t) \in \{x \in R^n \mid \|x - \tilde{x}\| \leq \epsilon$, *for some small* $\epsilon > 0\}$.

Based on the Liapùnov theory, the following lemma provides the existence of an equilibrium point for the neural dynamics defined by (11) as $s \to \infty$.

LEMMA 3.1. *If Problem (P) has an optimal solution, the neural dynamics defined by (11) has an equilibrium point, as* $s \to \infty$.

PROOF.

(i) If Problem (P) has an optimal solution with its objective value being f^*, the objective function $f(x)$ is bounded below over the feasible region F by f^*, i.e.,

$$f(x) \geq f^*, \tag{17}$$

for all $x \in F$ with equality hold, when x is optimal to Problem (P).

Now, by (2), (3), and (8), as $s \to \infty$, we have

$$E(x(t), s) = f(x(t)) + s \sum_{i=1}^{m} \Phi(g_i(x(t))) \begin{cases} = f(x(t)), & \text{if } x(t) \in F, \\ > f(x(t)), & \text{if } x(t) \notin F, \end{cases} \tag{18}$$

for $t \geq 0$ and $x(0) \in R^n$ being an arbitrary initial state. Thus, as $s \to \infty$, by (17), we have

$$E(x(t), s) \geq f(x(t)) \geq f^*, \tag{19}$$

with equality hold, when $x(t)$ is optimal to Problem (P). Therefore, as $s \to \infty$, if Problem (P) has an optimal solution, the optimal solution is a minimizer of $E(x(t), s)$.

(ii) For a minimizer of $E(x(t), s)$, since $s > 0$ and functions f, g_i, and Φ are convex, $E(x(t), s)$ is convex. Thus, by the necessary and sufficient conditions of optimality [20,21], $\tilde{x}$ is a minimizer of $E(x(t), s)$ if and only if

$$\nabla_{x(t)} E(\tilde{x}, s) = 0. \tag{20}$$

This result indicates that $\tilde{x}$ is a solution to the neural dynamics defined by (11). Therefore, by Definition 3.1, the minimizer $\tilde{x}$ of $E(x(t), s)$ is an equilibrium point of (11).

As a consequence of (i) and (ii), if Problem (P) has an optimal solution, the neural dynamics defined by (11) has an equilibrium point, as $s \to \infty$. ∎

We now prove the following lemma for the asymptotic stability of an equilibrium point.

LEMMA 3.2. *An equilibrium point of the neural dynamics defined by (11) is asymptotically stable.*

PROOF. Since $\eta > 0$, equation (11) results in

$$\nabla_{x(t)} E(x(t), s) = -\frac{1}{\eta} \dot{x}(t). \tag{21}$$

Moreover, since

$$\frac{dE(x(t), s)}{dt} = \left[\nabla_{x(t)} E(x(t), s)\right]^{\top} \dot{x}(t), \tag{22}$$

by substituting (21) for $\nabla_{x(t)} E(x(t), s)$, we have

$$\frac{dE(x(t), s)}{dt} = \left[-\frac{1}{\eta} \dot{x}(t)\right]^{\top} \dot{x}(t) = -\frac{1}{\eta} \|\dot{x}(t)\|^2. \tag{23}$$

Now, $\eta > 0$ and $\|\cdot\|^2 \geq 0$ imply

$$-\frac{1}{\eta}\|\dot{x}(t)\|^2 \leq 0. \tag{24}$$

Therefore, $\forall t \geq 0$,

$$\frac{dE(x(t),s)}{dt} \leq 0. \tag{25}$$

In particular, for an unstable state $x(t)$, since $\dot{x}(t) \neq 0$ implies that

$$\|\dot{x}(t)\|^2 > 0 \quad \text{and} \quad -\frac{1}{\eta}\|\dot{x}(t)\|^2 < 0, \tag{26}$$

we have

$$\frac{dE(x(t),s)}{dt} < 0. \tag{27}$$

Because f, g_i, and Φ are convex, $E(x(t),s)$ is convex. Furthermore, since $E(x(t),s)$ is assumed to increase unboundedly as $\|x(t)\| \to \infty$, $E(x(t),s)$ is bounded below. Therefore, by Theorem 3.4, the equilibrium point of the neural dynamics defined by (11) is asymptotically stable in the sense of Liapùnov. ∎

When the neural dynamics is asymptotically stable, the neural dynamics is obviously stable [22]. Therefore, we have a corollary for stability.

COROLLARY 3.2. *An equilibrium point of the neural dynamics defined by (11) is stable.*

On the other hand, as $s \to \infty$, an equilibrium point of the neural dynamics is an optimal solution to Problem (P), which is provided by the next lemma.

LEMMA 3.3. *If $\tilde{x}$ is an equilibrium point of the neural dynamics defined by (11) with an energy function $E(x(t),s)$ defined by (8), then $\tilde{x}$ is an optimal solution to Problem (P), as $s \to \infty$.*

PROOF. If the neural dynamics defined by (11) has an equilibrium point $\tilde{x}$, $\tilde{x}$ is a solution to equation (11), i.e.,

$$\dot{x}(t)|_{x(t)=\tilde{x}} = -\mu\nabla_{x(t)}E(\tilde{x},s) = 0, \tag{28}$$

for $t \geq 0$ and $x(0)$ be an arbitrary initial state.

Now, since $E(x(t),s)$ is convex, by the necessary and sufficient conditions of optimality, $\tilde{x}$ is a minimizer of $E(x(t),s)$. Moreover, for $\tilde{x}$ being a minimizer of $E(x(t),s)$, by Theorem 3.2, as $s \to \infty$, $\tilde{x}$ is optimal to Problem (P). This completes the proof. ∎

As a result of Lemmas 3.1 and 3.3, we know the following relationship between an optimal solution of Problem (P) and an equilibrium point of the neural dynamics defined by (11).

THEOREM 3.5. *As $s \to \infty$, an optimal solution of Problem (P) is an equilibrium point of the neural dynamics defined by (11), and vise versa.*

Finally, if s is sufficiently large, an equilibrium point of the neural dynamics defined by (11) provides a near-optimal solution to Problem (P).

THEOREM 3.6. *If $\tilde{x}$ is an equilibrium point of the neural dynamics defined by (11) with the neural energy $E(x(t),s)$ defined by (8), then, for a given $\epsilon > 0$, there exists a sufficiently large s such that the equilibrium point $\tilde{x}$ lies in $N(O,\epsilon)$, where $N(O,\epsilon) = \{x \in R^n \mid \|\tilde{x} - \bar{x}\| < \epsilon$, for some $\bar{x} \in O\}$ and O is the set of minimizers of (P).*

PROOF. Similar to the proof of Lemma 3.3, if $\tilde{x}$ is an equilibrium point of the neural dynamics defined by (11), $\tilde{x}$ is a minimizer of $E(x(t),s)$ by Definition 3.1, the convexity of $E(x(t),s)$, and the necessary and sufficient conditions of optimality. Then, by Corollary 3.1, for a sufficiently large s, a minimizer of $E(x(t),s)$ is a near-optimal solution to Problem (P). Therefore, for a given $\epsilon > 0$, there exists a sufficiently large s such that the equilibrium point $\tilde{x}$ lies in $N(O,\epsilon)$, where $N(O,\epsilon) = \{x \in R^n \mid \|x - \bar{x}\| < \epsilon$, for some $\bar{x} \in O\}$ and O is the set of minimizers of (P). ∎

4. NEURAL NETWORK CONFIGURATION

In order to solve convex programming problems with equality constraints by using the proposed neural network, the key is to define the connection weights and neuron activation functions.

By analyzing the neural dynamics defined by equation (11), we have

$$\begin{aligned}\dot{x}(t) &= -\eta \nabla_{x(t)} E(x(t), s) \\ &= -\eta \left\{ \nabla_{x(t)} f(x(t)) + s \sum_{i=1}^{m} \nabla_{x(t)} \Phi\left(g_i\left(x(t)\right)\right) \right\} \\ &= -\eta \left\{ \nabla_{x(t)} f(x(t)) + s \sum_{i=1}^{m} \phi\left(g_i\left(x(t)\right)\right) \nabla_{x(t)} g_i(x(t)) \right\} \\ &= -\eta \nabla_{x(t)} f(x(t)) - \eta s \sum_{i=1}^{m} \phi\left(g_i\left(x(t)\right)\right) \nabla_{x(t)} g_i\left(x(t)\right).\end{aligned} \tag{29}$$

Then, similar to the McCulloch-Pitts neuron model in [25], a neural network for solving convex programming problems will have two types of neurons. The first type has a neuron potential $x(t)$ with coefficients of $x(t)$ in $g(x(t))$ as connection weights, some negative values of constant terms of $g(x(t))$ as thresholds, and the derivative of the penalty function Φ as neuron activation functions. The output of each neuron is computed as the derivative of the penalty function evaluated at the value of $g(x(t))$. By taking the output values of the first type of neurons as inputs, the second type has coefficients of $\phi(g(x(t)))$ in $-\eta s \sum_{i=1}^{m} \phi(g_i(x(t)))\nabla_{x(t)} g_i(x(t))$ as connection weights, values of $\eta \nabla_{x(t)} f(x(t))$ as thresholds, and integrators as neuron activation functions to provide $x(t)$ for the input of the first type of neurons in the next iteration. This process is repeated, and the solutions computed by the neural network can be obtained by measuring the output voltage of the second type of neurons in the corresponding neural circuit after the neural network reaches

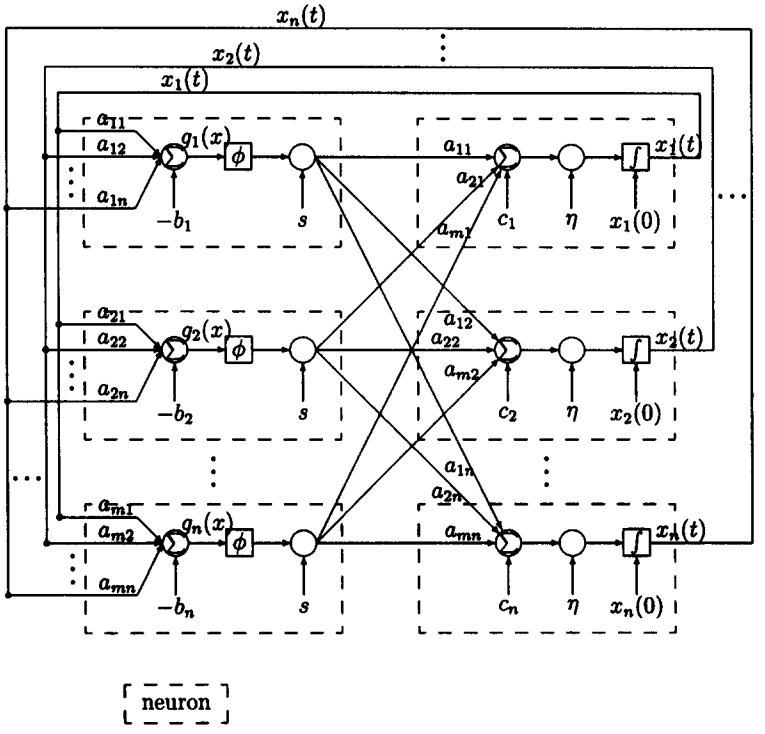


Figure 1. Neural network structure for linear programming problems.

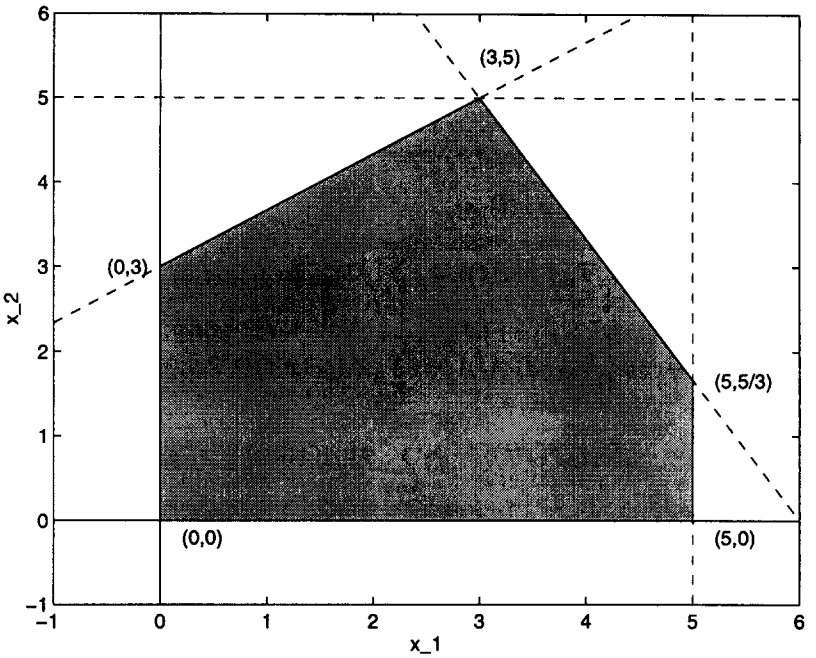


Figure 2. Feasible region of illustrative examples.

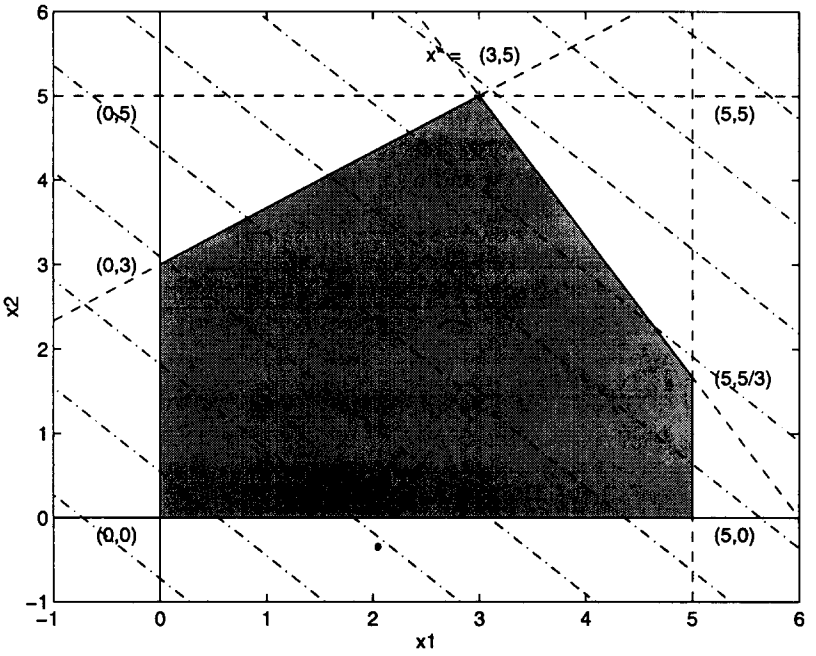


Figure 3. Contour lot of function f_1 over region F.

its equilibrium. Then, by Theorem 3.6, the equilibrium point of the neural network will provide a near-optimal solution to the corresponding convex programming problem, if s is sufficiently large.

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

$$(\mathrm{P}_{\leq}) \qquad \begin{aligned} &\min f(x), \\ &\text{s.t.} \quad h(x) \leq 0, \\ &\quad x \in X, \end{aligned} \tag{30}$$

where $f : R^n \to R$, $h : R^n \to R^m$, and $X \subseteq R^n$, we can transform the inequality constraints into

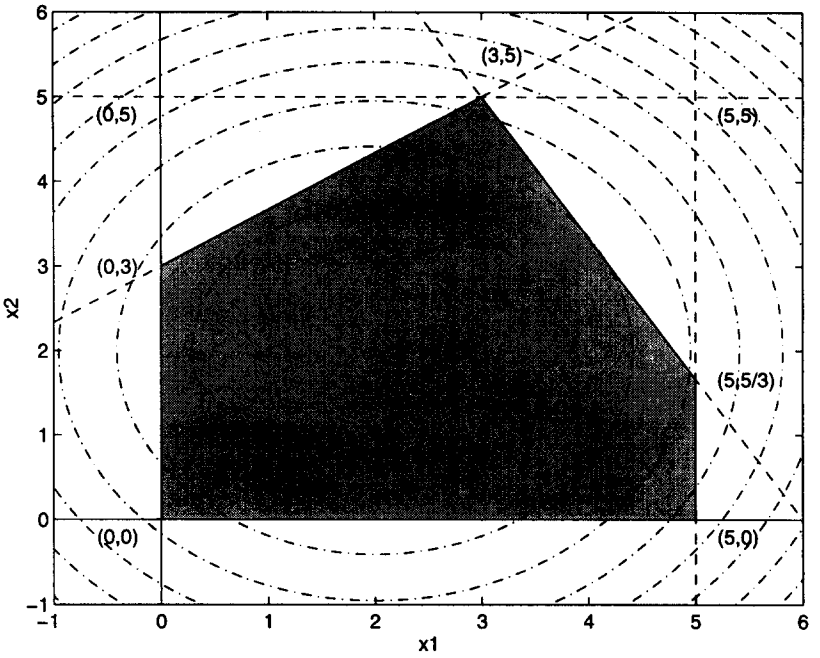


Figure 4. Contour plot of function f_2 over region F.

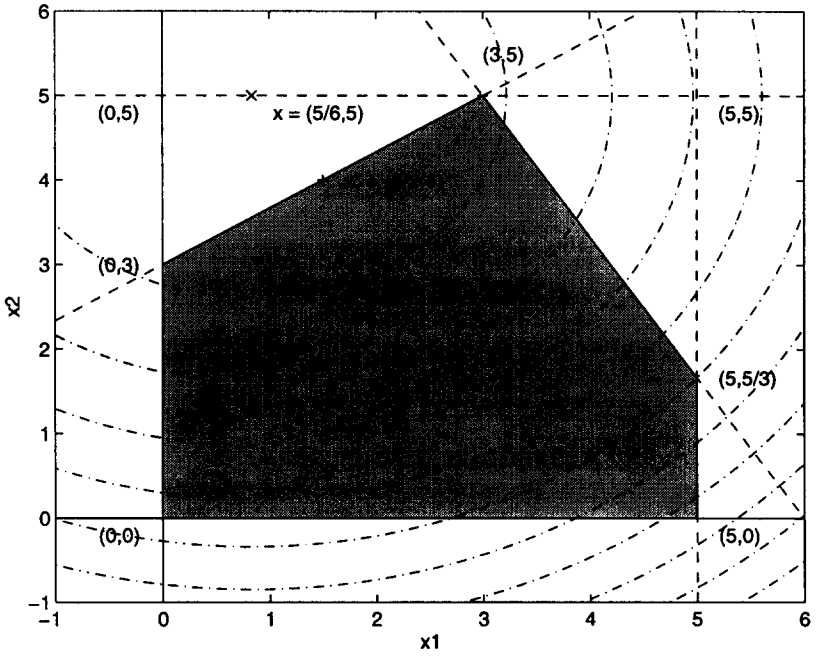


Figure 5. Contour plot of function f_3 over region F.

equality constraints by adding nonnegative slack variables, i.e.,

$$g(x) = h(x) + x_s = 0, \tag{31}$$

where $x_s \in R^m$ and $x_s \geq 0$. Then, we can follow the same procedure to develop a neural network for finding optimal solutions.

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 [15]. This means that the input signals of an integrator are integrated but cannot drive the output x_j beyond

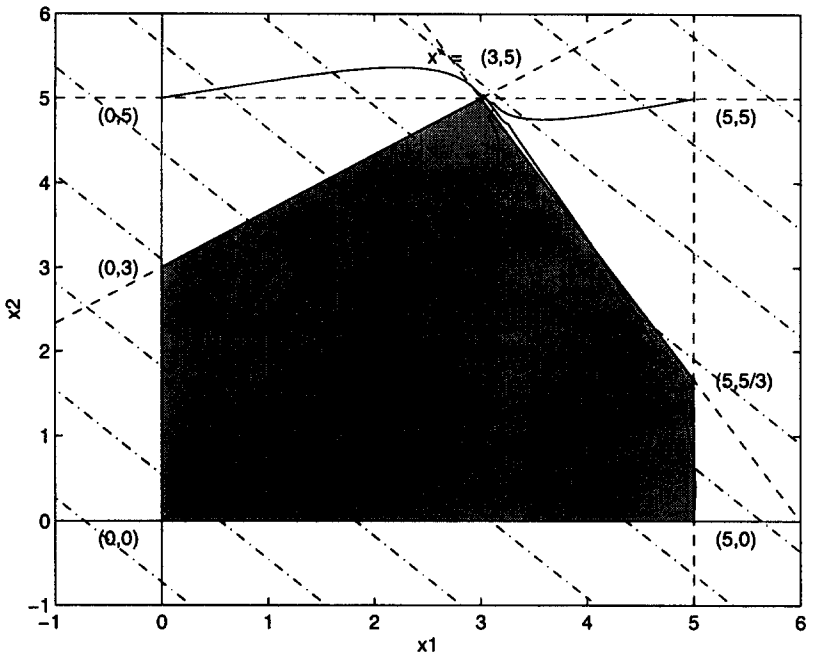


Figure 6. State trajectories of $x_1(t)$ and $x_2(t)$ for Problem (P_1).

the specified limits. In such an approach, all box constraints are "hard", i.e., the constraints must not be violated either at the final solution or during the optimization process.

An alternative approach is to introduce an unbounded variable u_j, for $j = 1, 2, \ldots, n$, that provides the nonlinear transformations

$$x_j = h_j(u_j), \tag{32}$$

such as, for a given $\gamma > 0$,

$$x_j = x_j^{\min} + \frac{x_j^{\max} - x_j^{\min}}{1 + e^{-\gamma u_j}}. \tag{33}$$

The following linear programming problem is used to illustrate the configuration of such a neural network:

$$\text{(LP)} \qquad \begin{aligned} &\min c^\top x, \\ &\text{s.t.} \quad Ax = b, \\ &\quad\ x \geq 0, \end{aligned} \tag{34}$$

where $c = [c_j] \in R^n$, $A = [a_{ij}] \in R^{m \times n}$, $b = [b_i] \in R^m$, and $x = [x_j] \in R^n$. Let

$$f(x(t)) = c^\top x(t), \qquad \nabla_x f(x(t)) = c, \tag{35}$$

$$g(x(t)) = Ax(t) - b, \qquad \text{and} \tag{36}$$

$$F = \{x \in R^n \mid g(x) = 0 \text{ and } x \geq 0\}. \tag{37}$$

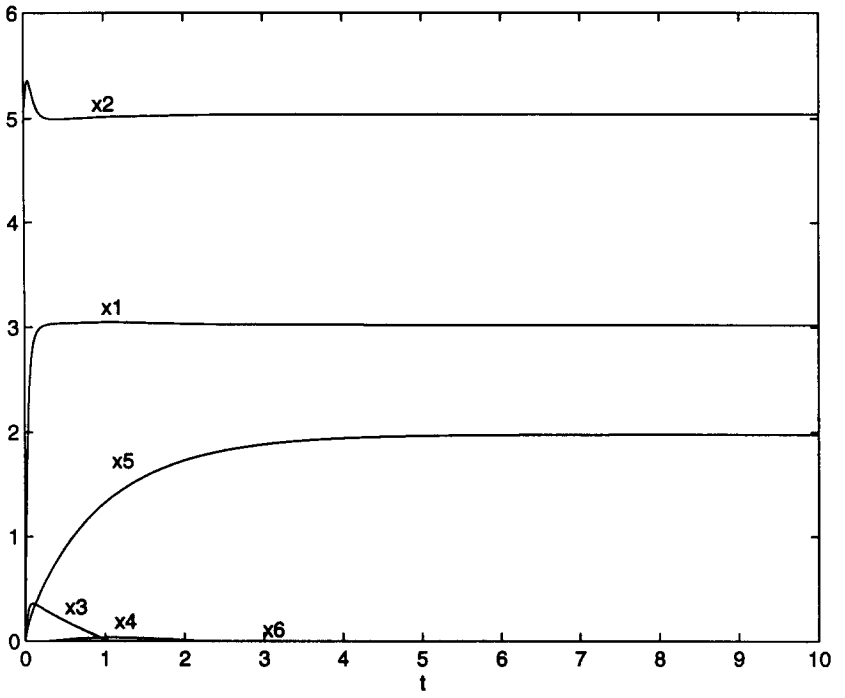
Then by choosing constants s, and η, and penalty function

$$\Phi(g_i(x(t))) = \ln \cosh g_i(x(t)), \tag{38}$$

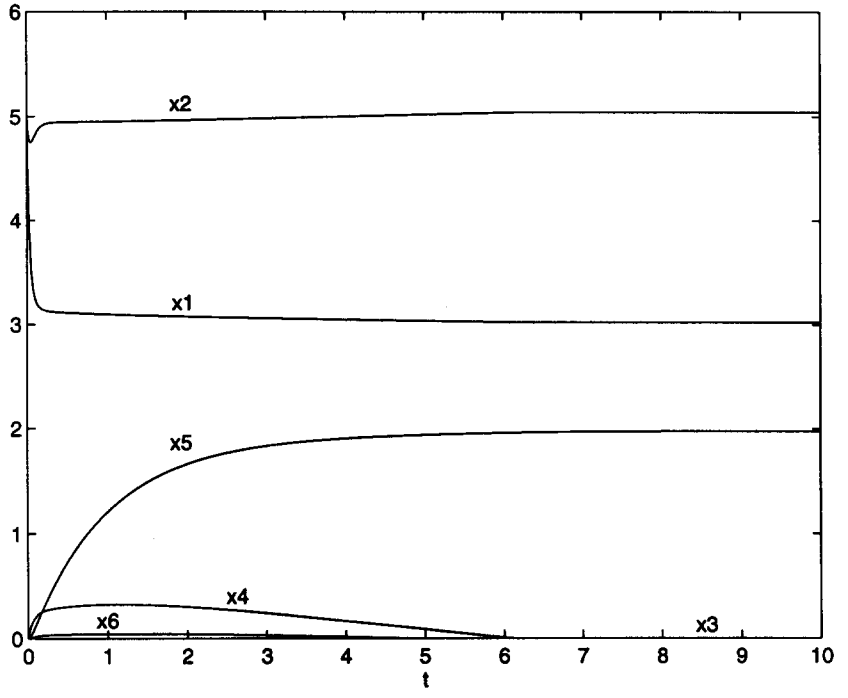
with gradient function

$$\phi(g_i(x(t))) = \tanh g_i(x(t)), \tag{39}$$

we can develop a neural network for solving linear programming problems. Figure 1 shows such a neural network architecture.



(a) $x(0) = x^1$.

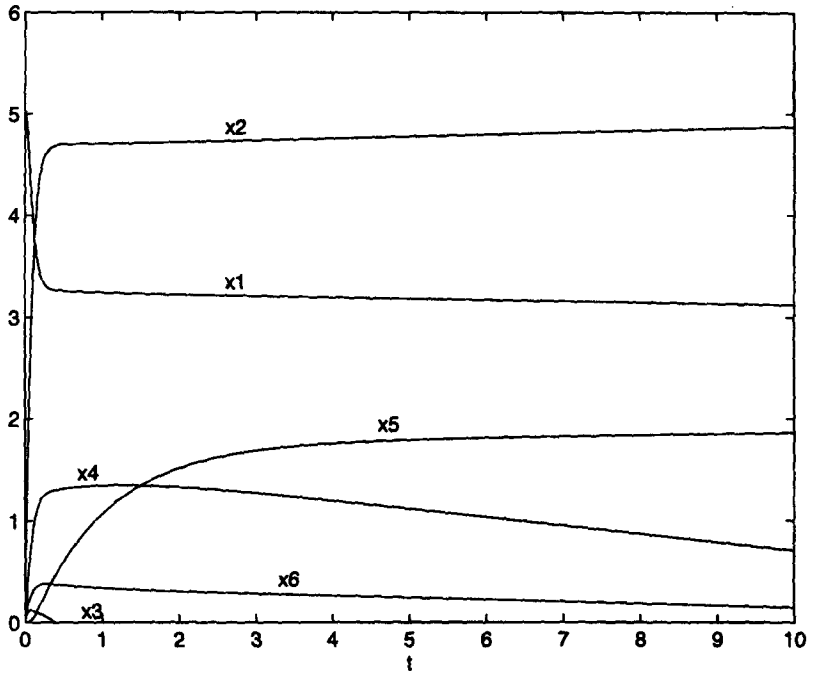


(b) $x(0) = x^2$.

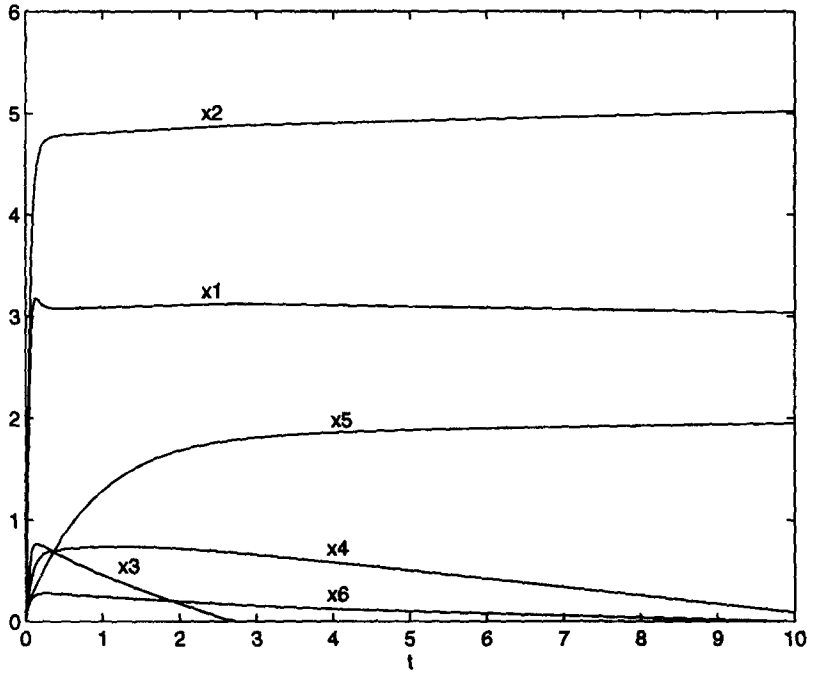
Figure 7. State trajectories of Problem (P_1) for different initial states, $t \in [0, 10]$.

5. NUMERICAL EXAMPLES

To demonstrate the behavior and properties of the proposed neural network, we conduct experiments on one linear and two convex quadratic programming problems. All three problems share the following set of constraints:



(c) $x(0) = x^3$.



(d) $x(0) = x^4$.

Figure 7. (cont.)

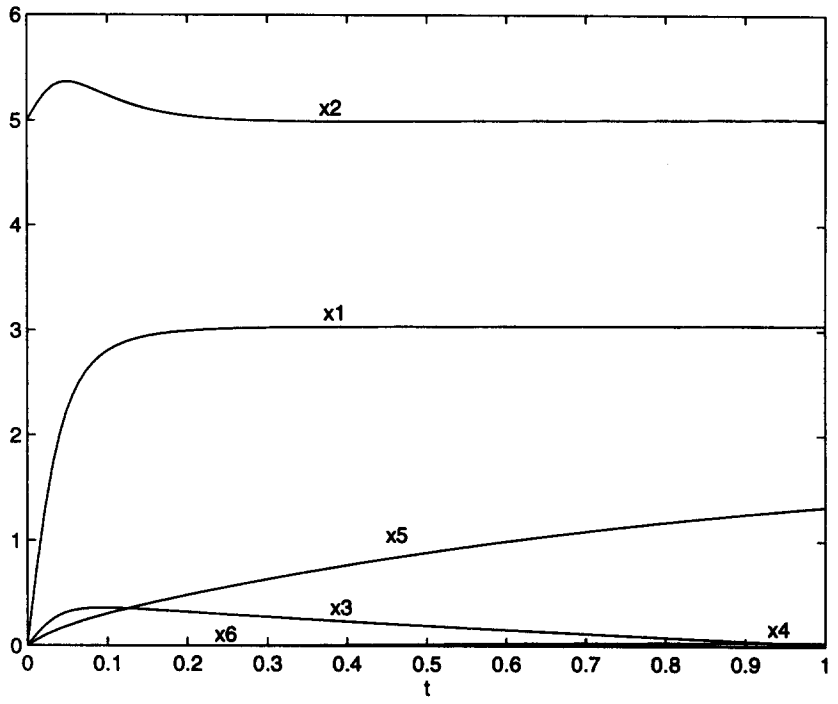
$$5x_1 + 3x_2 \leq 30, \tag{40}$$

$$2x_1 - 3x_2 \geq -9, \tag{41}$$

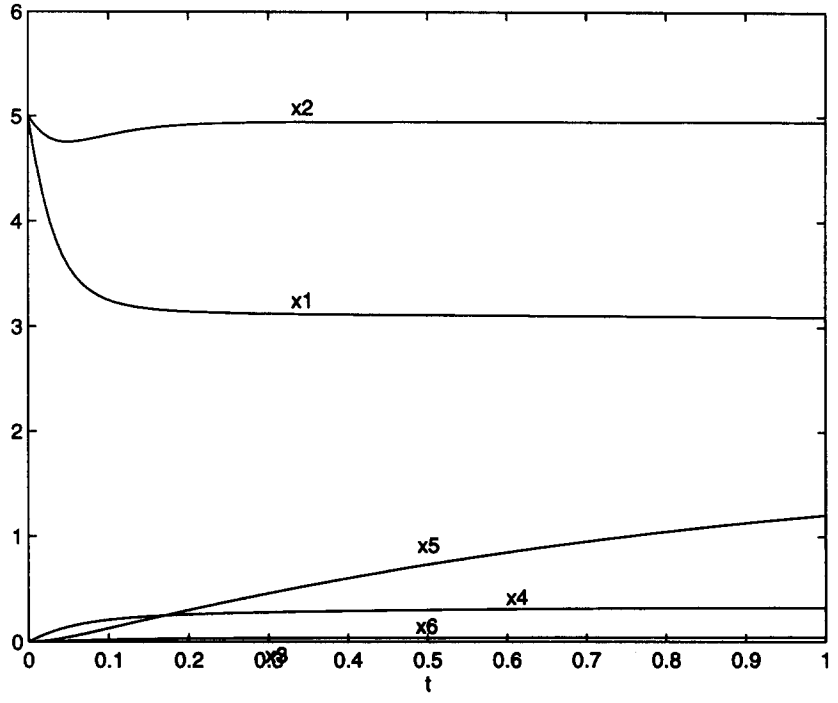
$$x_1 \leq 5, \tag{42}$$

$$x_2 \leq 5, \tag{43}$$

$$x_1 \geq 0, \qquad x_2 \geq 0.$$



(a) $x(0) = x^1$.



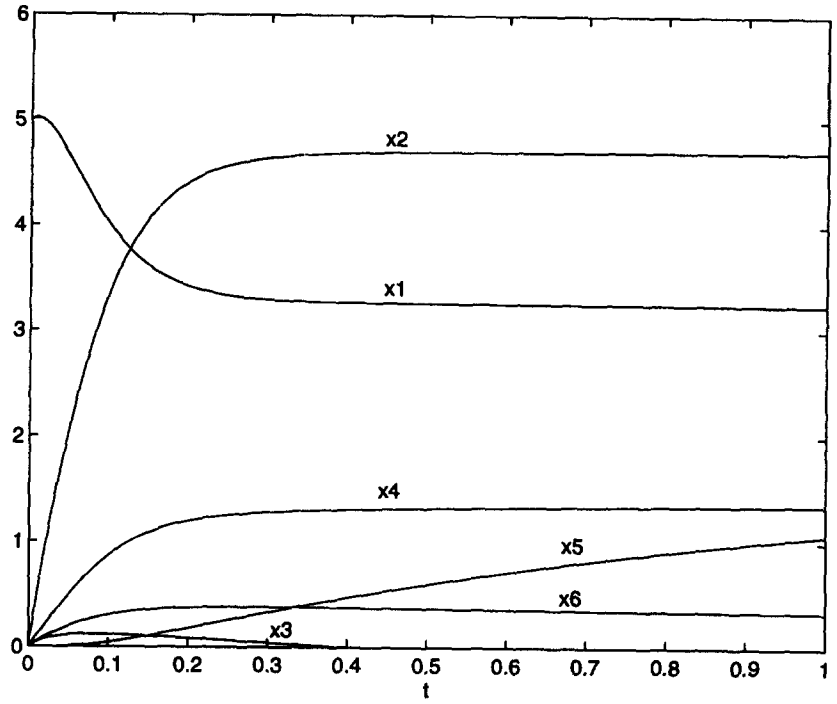
(b) $x(0) = x^2$.

Figure 8. State trajectories of Problem (P_1) for different initial states, $t \in [0, 1]$.

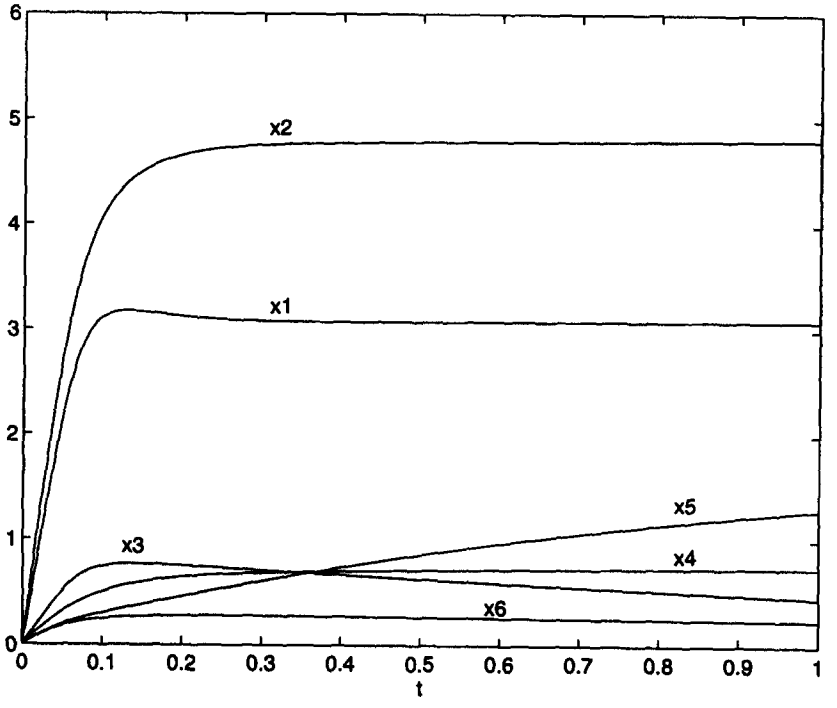
Let F represent a set of feasible solutions defined by these constraints, i.e.,

$$F = \{x \in R^2 \mid 5x_1 + 3x_2 \leq 30,\ 2x_1 - 3x_2 \geq -9,\ x_1 \leq 5,\ x_2 \leq 5,\ x_1 \geq 0, \text{ and } x_2 \geq 0\}. \quad (44)$$

Figure 2 shows the feasible region with five extreme points: (0,0), (5,0), (5,5/3), (3,5), and (0,3).



(c) $x(0) = x^3$.



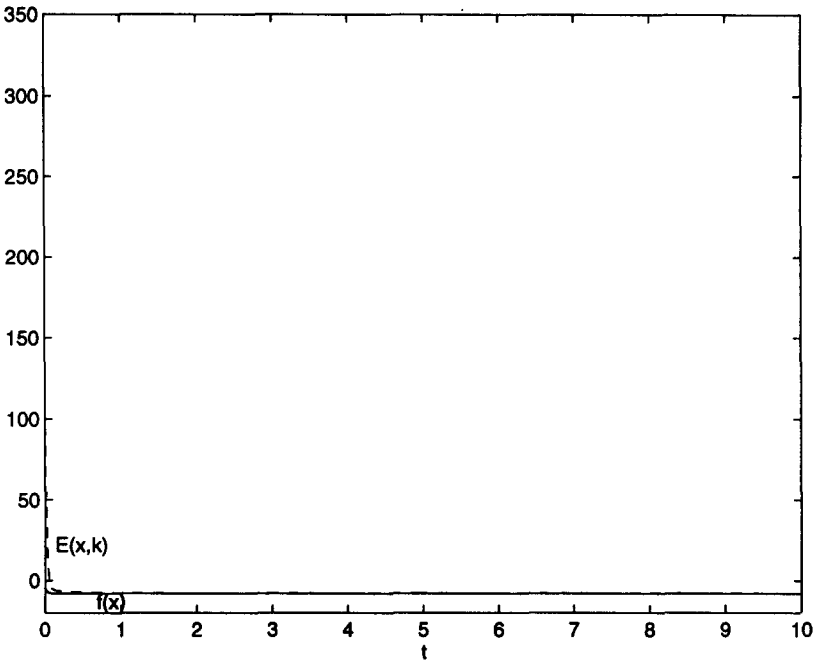
(d) $x(0) = x^4$.

Figure 8. (cont.)

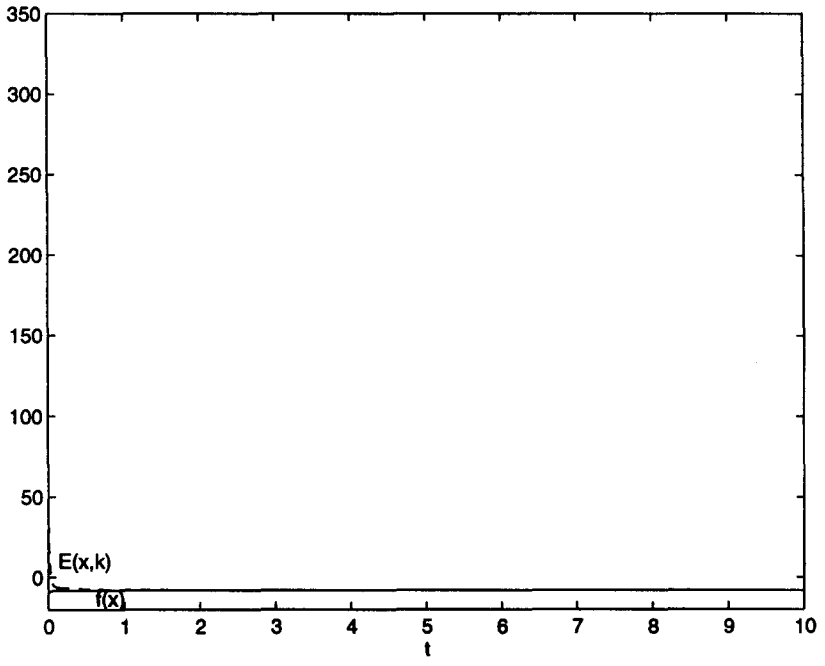
By letting $x = [x_1, x_2]^\top$, the objective of the linear programming Problem (P_1) is to minimize

$$f_1(x) = -x_1 - x_2, \tag{45}$$

subject to $x \in F$. Obviously, the optimal solution of this linear programming problem occurs at the extreme point $(3, 5)$. The contour plot of this problem is shown by Figure 3. The two



(a) $x(0) = x^1$.



(b) $x(0) = x^2$.

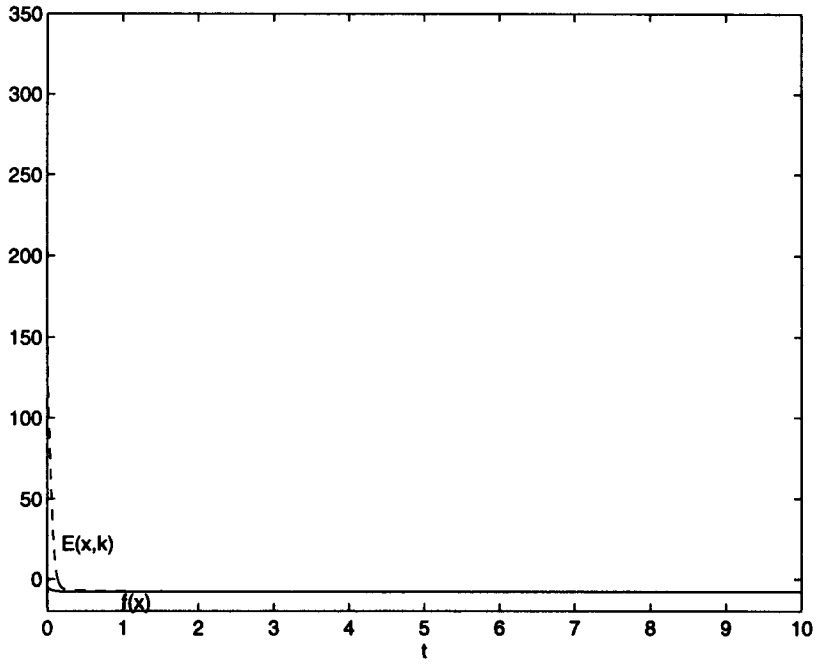
Figure 9. Energy and objective trajectories of Problem (P_1) for different initial states, $t \in [0, 10]$.

objective functions of the quadratic programming Problems (P_2) and (P_3) are to minimize

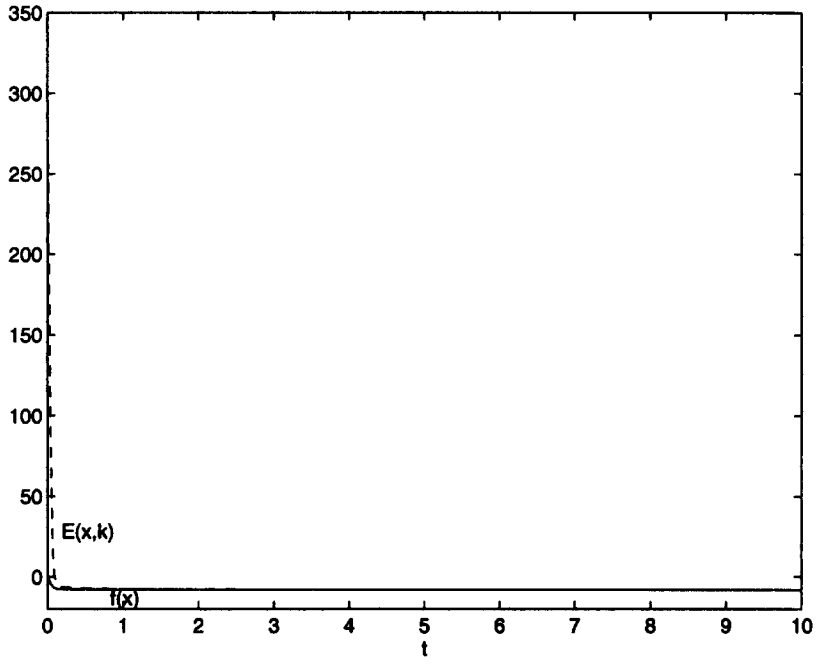
$$f_2(x) = \frac{1}{2}(x_1 - 2)^2 + \frac{1}{2}(x_2 - 2)^2 \tag{46}$$

and

$$f_3(x) = \frac{1}{2}\left(x_1 - \frac{5}{6}\right)^2 + \frac{1}{2}(x_2 - 5)^2, \tag{47}$$



(c) $x(0) = x^3$.

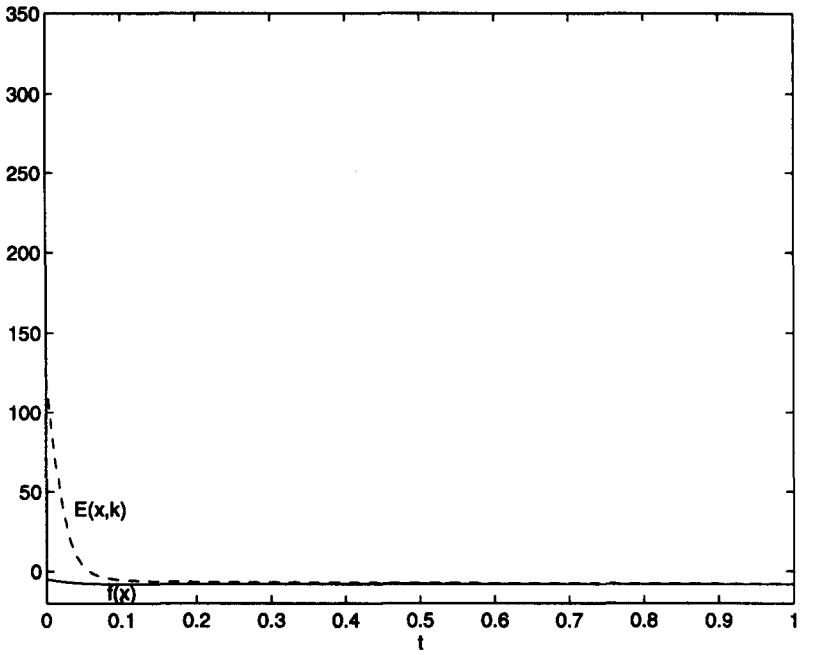


(d) $x(0) = x^4$.

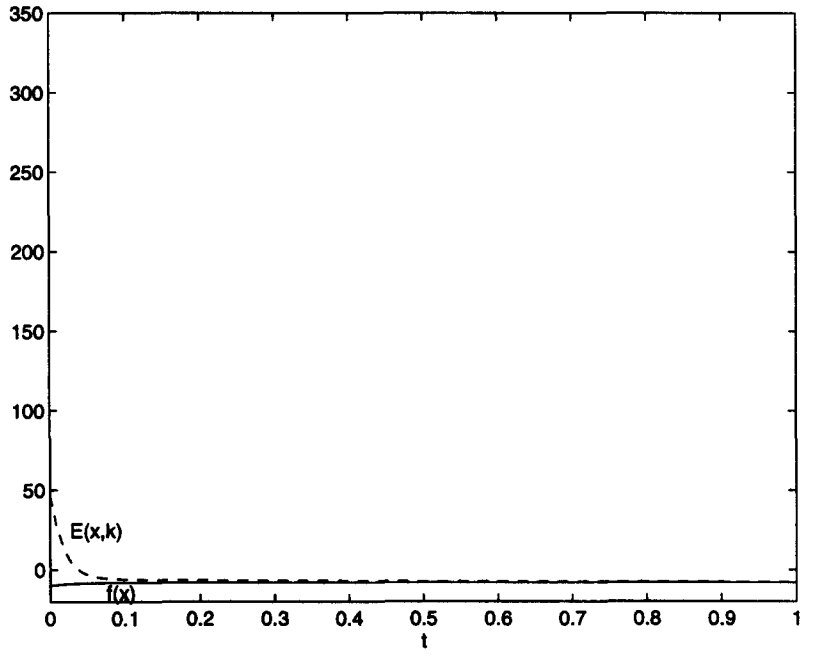
Figure 9. (cont.)

respectively, subject to $x \in F$. Without considering the constraints, function f_2 has a minimum point at (2,2), which is a feasible point of (P_2); function f_3 has a minimum point at (5/6, 5), which is an infeasible point of (P_3). Therefore, the former has an optimal solution in the interior of F; and the latter has an optimal solution on the boundary of feasible region F. The contour plots of these objective functions are shown in Figures 4 and 5, respectively.

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,



(a) $x(0) = x^1$.

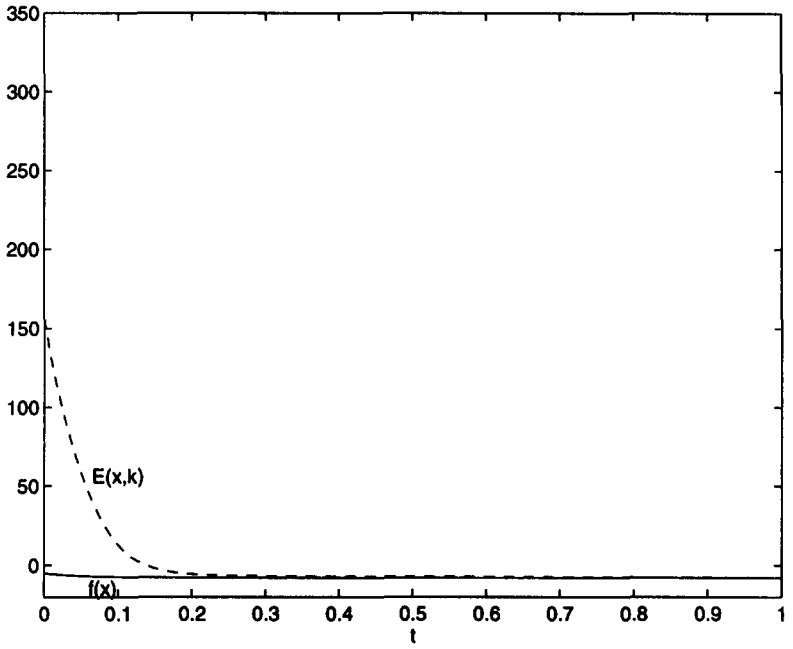


(b) $x(0) = x^2$.

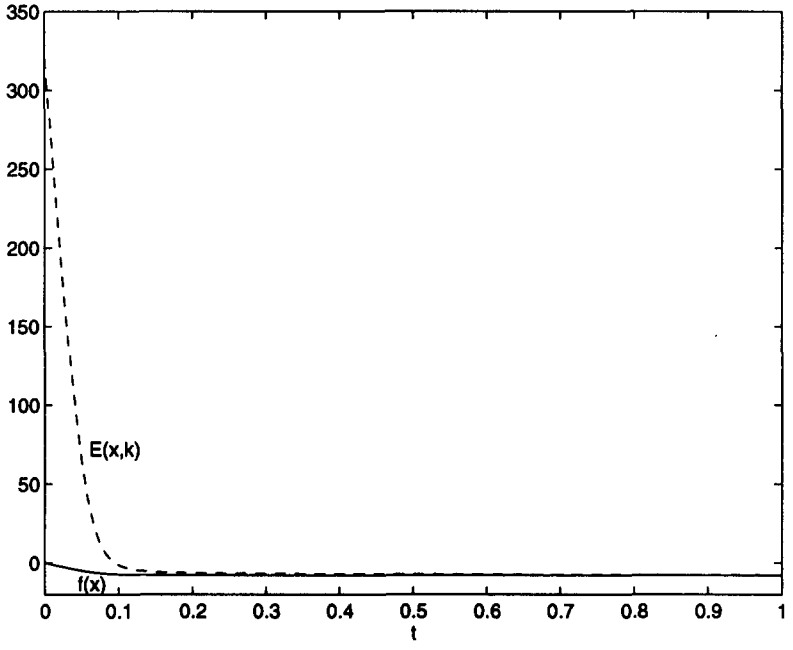
Figure 10. Energy and objective trajectories of Problem (P_1) for different initial states, $t \in [0, 1]$.

and x_6, i.e.,

$$\begin{aligned} g_1(x) &= 5x_1 + 3x_2 + x_3 - 30 = 0, \\ g_2(x) &= 2x_1 - 3x_2 - x_4 + 9 = 0, \\ g_3(x) &= x_1 + x_5 - 5 = 0, \\ g_4(x) &= x_2 + x_6 - 5 = 0, \\ x_1, x_2, \ldots, x_6 &\geq 0. \end{aligned} \tag{48}$$



(c) $x(0) = x^3$.



(d) $x(0) = x^4$.

Figure 10. (cont.)

Then, the state variables of the neural networks corresponding to those problems are $x(t) = [x_1(t), x_2(t), \ldots, x_6(t)]^\top$. All numerical experiments start with the following four initial states (points):

$$x^1 = [0, 5, 0, 0, 0, 0]^\top,$$
$$x^2 = [5, 5, 0, 0, 0, 0]^\top,$$
$$x^3 = [5, 0, 0, 0, 0, 0]^\top,$$
$$x^4 = [0, 0, 0, 0, 0, 0]^\top.$$

Table 1. Solution quality of Problem (P_1) at $t = 10$ for different penalty parameters.

s	$x(10)$	$\|x(10) - x^*\|$
1	$(3.2482, 5.4235, 0, 0, 1.1457, 0)^T$	0.9853
10	$(3.0214, 5.0435, 0, 0, 1.9779, 0)^T$	0.0533
50	$(3.0032, 5.0081, 0, 0, 1.9956, 0)^T$	0.0098
100	$(3.0039, 5.0053, 0, 0, 1.9978, 0)^T$	0.0069
500	$(3.0002, 5.0007, 0, 0, 1.9996, 0)^T$	0.0009

From Figures 3–5, we see that, in terms of x_1 and x_2, the four points are around the optimal solutions in different direction. For those problems, let $\eta = 1$, $s = 10$,

$$\Phi\left(g_i\left(x(t)\right)\right) = 10 \ln \cosh \frac{g_i(x(t))}{10} \tag{49}$$

and

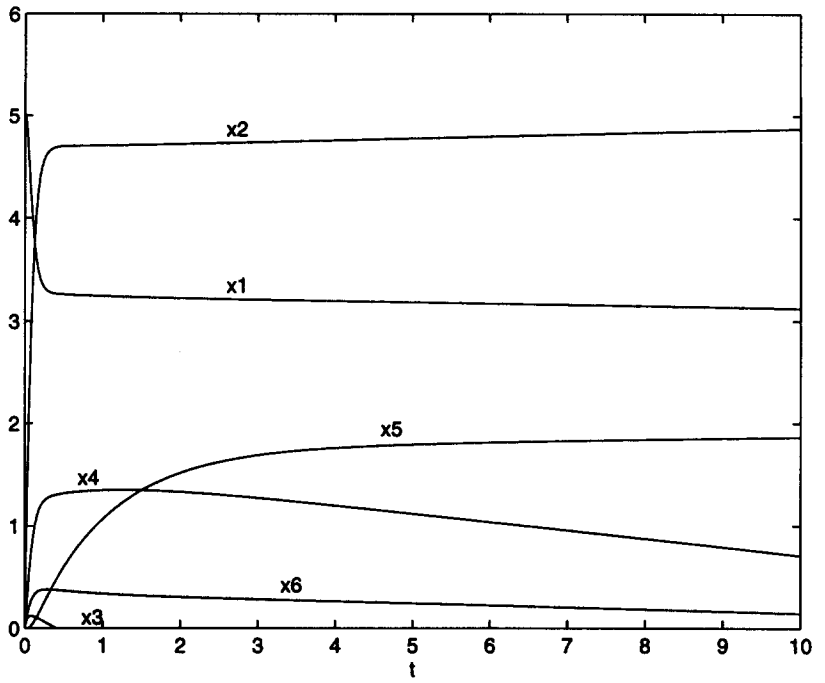
$$\phi\left(g_i\left(x(t)\right)\right) = \frac{\partial \Phi\left(g_i\left(x(t)\right)\right)}{\partial g_i(x(t))} = \tanh \frac{g_i(x(t))}{10}, \tag{50}$$

for $i = 1, 2, 3$, and 4. The following energy function is formulated for solving the linear programming problem for $t \geq 0$:

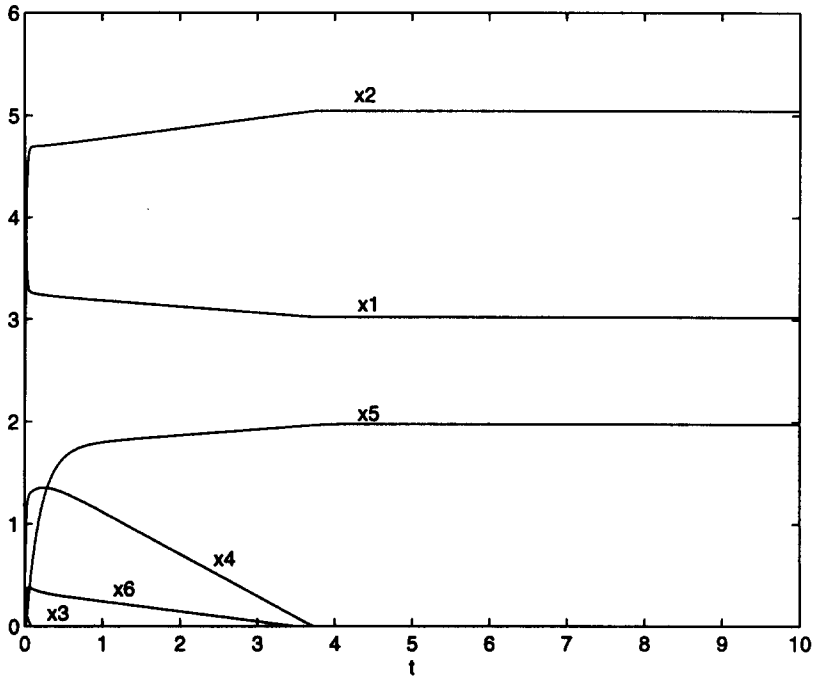
$$\begin{aligned} E(x(t), s) &= f_1(x(t)) + s \sum_{i=1}^{4} \Phi\left(g_i\left(x(t)\right)\right) \\ &= f_1(x(t)) + s \sum_{i=1}^{4} 10 \ln \cosh \frac{g_i(x(t))}{10} \\ &= -x_1(t) - x_2(t) + 100 \left\{\ln \cosh \left[0.1\left(5x_1(t) + 3x_2(t) + x_3(t) - 30\right)\right]\right. \\ &\quad + \ln \cosh \left[0.1(2x_1(t) - 3x_2(t) - x_4(t) + 9)\right] + \ln \cosh \left[0.1(x_1(t) + x_5(t) - 5)\right] \\ &\quad \left. + \ln \cosh \left[0.1\left(x_2(t) + x_6(t) - 5\right)\right]\right\}. \end{aligned} \tag{51}$$

Hence, by taking the derivative of energy function $E(x(t), s)$ with respect to $x(t)$, we have the following system of differential equations representing the neural dynamics:

$$\begin{aligned} \dot{x}(t) &= -\eta \nabla_{x(t)} E(x(t), s) \\ &= -\eta \left\{ \nabla_{x(t)} f_1(x(t)) + s \sum_{i=1}^{4} \phi(g_i(x(t))) \nabla_{x_i(t)} g_i(x(t)) \right\} \\ &= -\eta \nabla_{x(t)} f_1(x(t)) - \eta s \sum_{i=1}^{4} \phi(g_i(x(t))) \nabla_{x_i(t)} g_i(x(t)) \\ &= \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - 10 \begin{bmatrix} 5 \tanh \dfrac{g_1(x(t))}{10} + 2 \tanh \dfrac{g_2(x(t))}{10} + \tanh \dfrac{g_3(x(t))}{10} \\ 3 \tanh \dfrac{g_1(x(t))}{10} - 3 \tanh \dfrac{g_3(x(t))}{10} + \tanh \dfrac{g_4(x(t))}{10} \\ \tanh \dfrac{g_1(x(t))}{10} \\ -\tanh \dfrac{g_2(x(t))}{10} \\ \tanh \dfrac{g_3(x(t))}{10} \\ \tanh \dfrac{g_4(x(t))}{10} \end{bmatrix}. \end{aligned} \tag{52}$$



(a) $\eta = 1$.



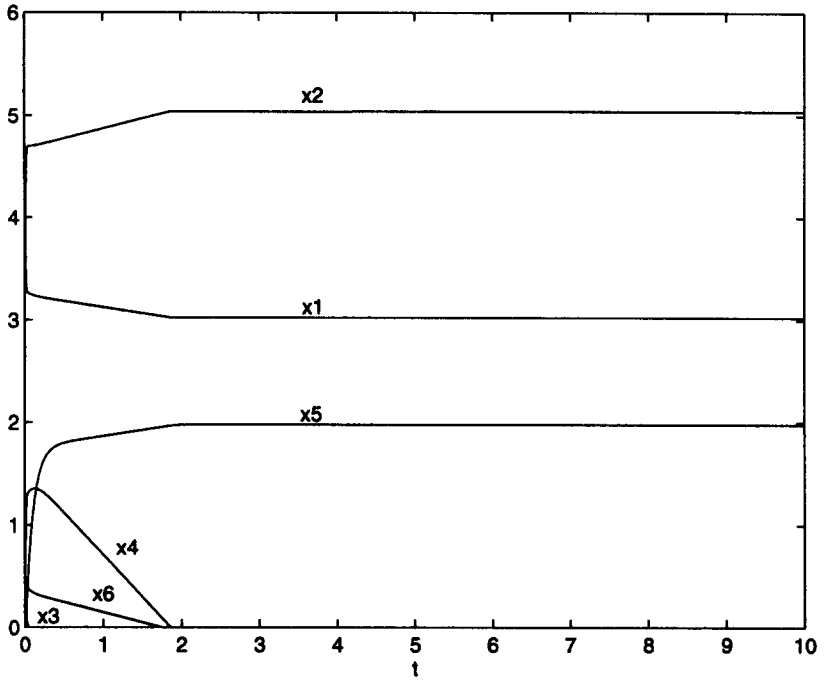
(b) $\eta = 5$.

Figure 11. State trajectories of Problem (P_1) for different learning constants with $x(0) = x^3$, $t \in [0, 10]$.

Therefore,

$$\dot{x}_1(t) = 1 - 10\left(5\tanh\frac{g_1(x(t))}{10} + 2\tanh\frac{g_2(x(t))}{10} + \tanh\frac{g_3(x(t))}{10}\right), \tag{53}$$

$$\dot{x}_2(t) = 1 - 10\left(3\tanh\frac{g_1(x(t))}{10} - 3\tanh\frac{g_3(x(t))}{10} + \tanh\frac{g_4(x(t))}{10}\right), \tag{54}$$



(c) $\eta = 10$.

Figure 11. (cont.)

$$\dot{x}_3(t) = -10 \tanh \frac{g_1(x(t))}{10}, \tag{55}$$

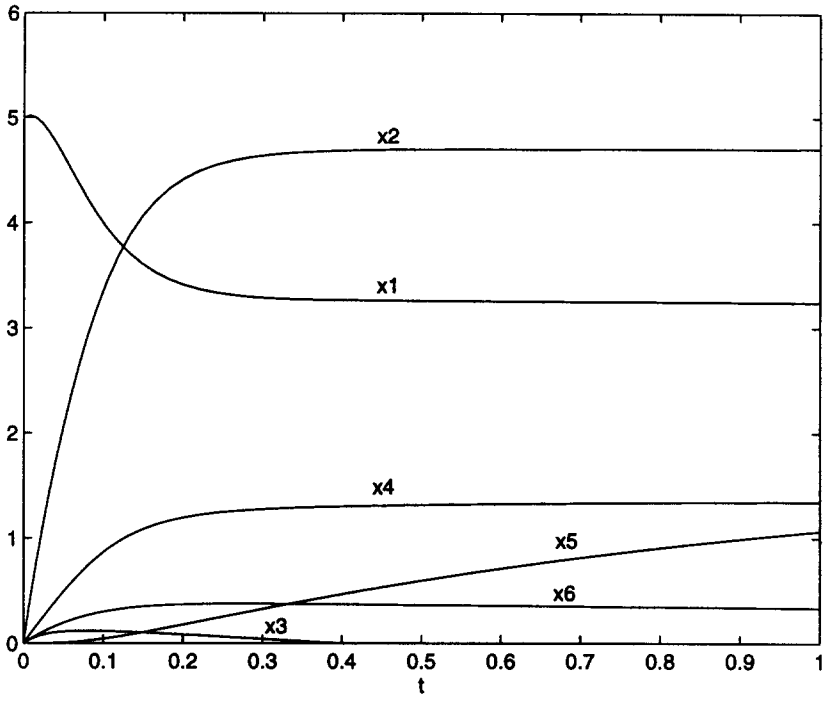
$$\dot{x}_4(t) = 10 \tanh \frac{g_2(x(t))}{10}, \tag{56}$$

$$\dot{x}_5(t) = -10 \tanh \frac{g_3(x(t))}{10}, \tag{57}$$

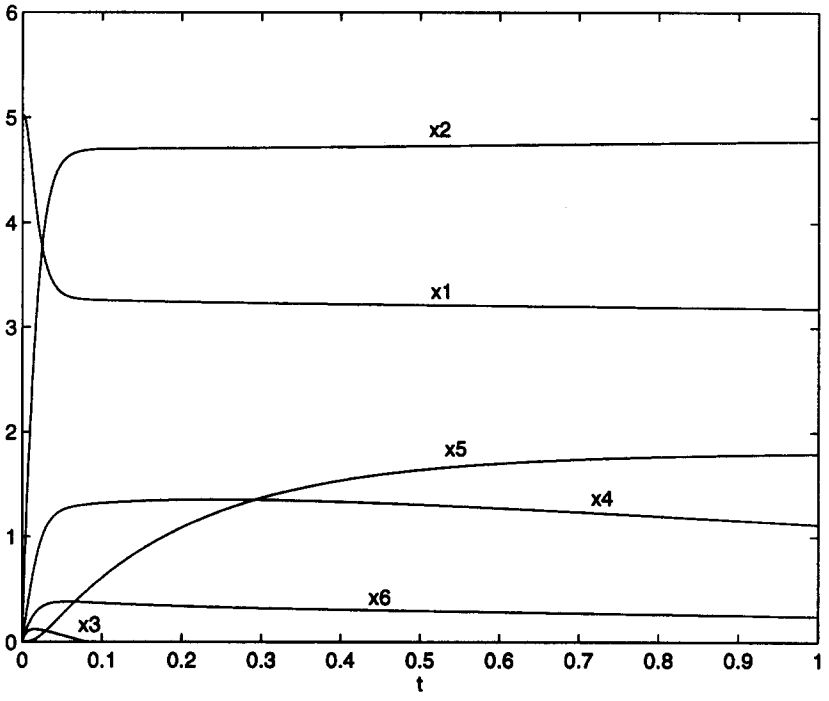
$$\dot{x}_6(t) = -10 \tanh \frac{g_4(x(t))}{10}. \tag{58}$$

Figure 6 gives trajectories of states $x_1(t)$ and $x_2(t)$ over the feasible region for Problem (P_1). The numerical method used to generate the trajectories is a fourth-order Runge-Kutta method [26]. In the figure, all four initial states converge to the optimal solution, the equilibrium point of the neural dynamics, $(x_1^*, x_2^*) = (3, 5)$. Figures 7 and 8 show the state trajectories from each initial state for $t \in [0, 10]$ and $t \in [0, 1]$, respectively. In the figures, both variables $x_1(t)$ and $x_2(t)$ converge to the neighborhood of the equilibrium point at $t = 0.4$. Figures 9 and 10 give trajectories of objective function $f_1(x(t))$ and energy function $E(x(t), s)$ of Problem (P_1) for $t \in [0, 10]$ and $t \in [0, 1]$, respectively. In the figures, the value of the objective function converges to its minimum value at $t = 0.2$, when the network's energy function gradually reaches its minimum energy. Table 1 gives distances between the network's states at $t = 10$ and the optimal solution $x^* = [3, 5, 0, 0, 2, 0]^T$ for various penalty parameters with $x(0) = x^1$.

In the table, the distance of the solution obtained at $t = 10$ from the optimal one depends on the magnitude of the penalty parameter. Figures 11 and 12 show the state trajectories, when $x(0) = x^3$, with different learning coefficient constants η for $t \in [0, 10]$ and $t \in [0, 1]$, respectively. We can see that the state trajectories converge to the equilibrium point faster, when a large learning coefficient constant is applied. Figures 13 and 14 show the state trajectories for different penalty functions $\Phi(g_i(x(t)))$, starting from $x(0) = x^3$, for $t \in [0, 10]$ and $t \in [0, 1]$, respectively. These figures show that the neural network with penalty function $\Phi(g_i(x(t))) = 1/2(g_i(x(t)))^2$ (Figures 13a and 14a) converges to the network's equilibrium point a little faster at the beginning



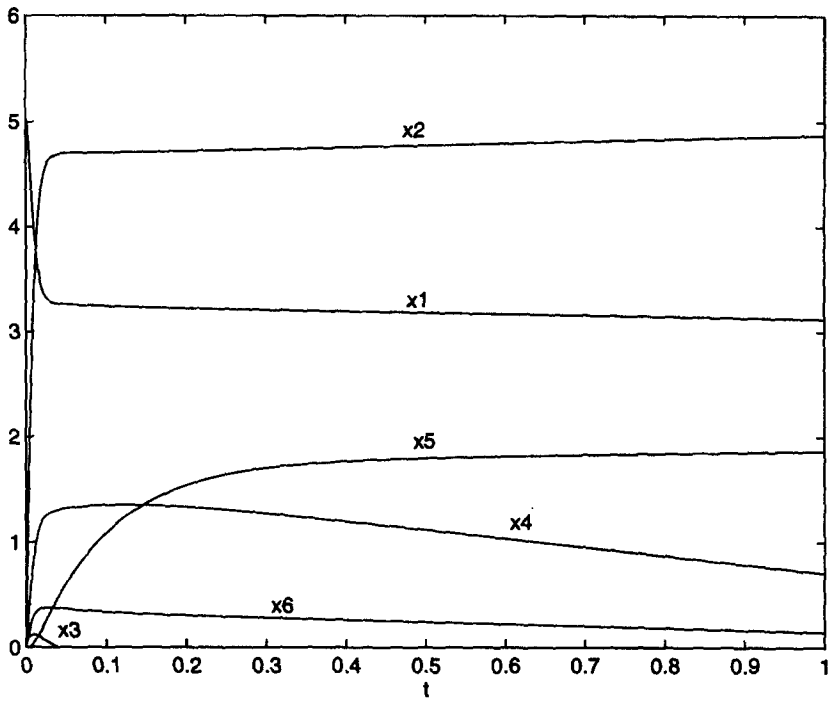
(a) $\eta = 1$.



(b) $\eta = 5$.

Figure 12. State trajectories of Problem (P_1) for different learning constants with $x(0) = x^3$, $t \in [0, 1]$.

of the computer simulation than the one with penalty function $\Phi(g_i(x(t))) = 10 \ln \cosh g_i(x(t))/10$ (Figures 13b and 14b). Therefore, as a result of the table and these figures, the quality of solution and the convergence speed of the neural network depend on the penalty parameter s, the learning coefficient η, and the penalty function $\Phi(g_i(x(t)))$.



(c) $\eta = 10$.

Figure 12. (cont.)

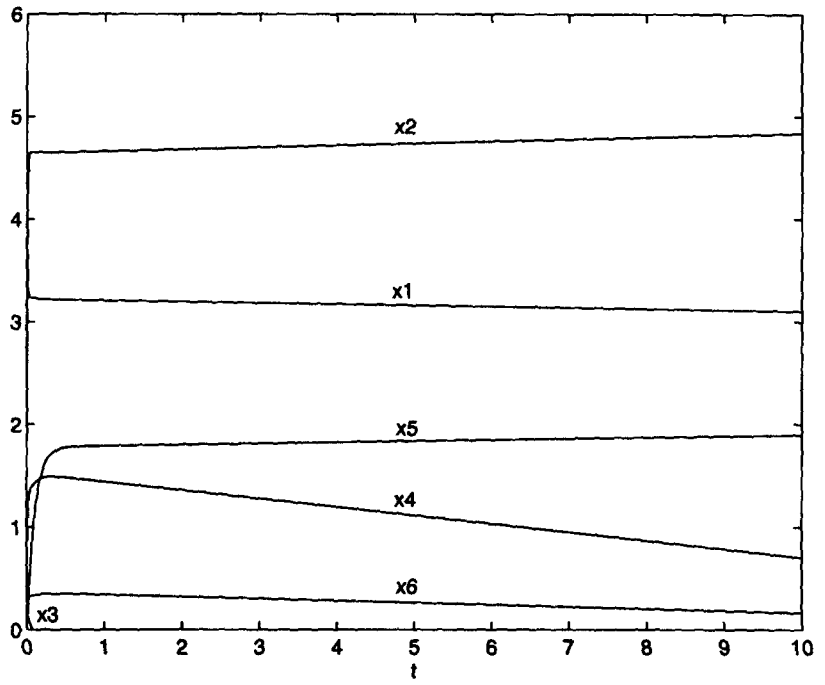
By using the same initial setup as the one for Problem (P_1), let the quadratic programming problem with objective function $f_2(x)$ be Problem (P_2). Figure 15 gives the state trajectories of a neural network for Problem (P_2). In the figure, each initial state converges to the equilibrium point of the neural network, which corresponds to the optimal solution of Problem (P_2). By letting the quadratic programming problem with objective function $f_3(x)$ be (P_3), Figure 16 gives the state trajectories of a neural network for Problem (P_3). In the figure each initial state also converges to the equilibrium point of the neural network, which corresponds to the optimal solution of Problem (P_3).

6. CONCLUSION

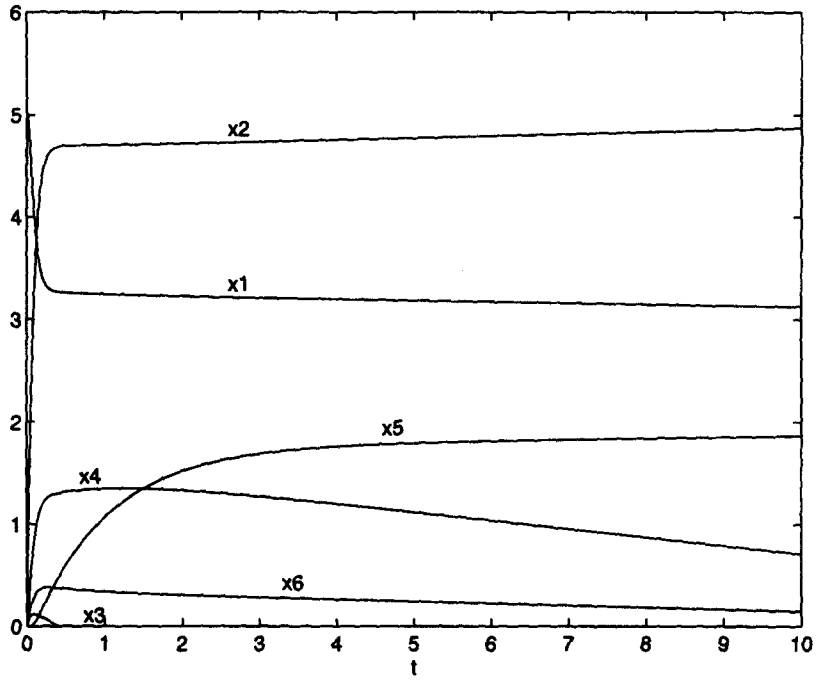
In this paper, a special neural network has been proposed for solving convex programming problems with equality constraints. It is a generalization of the neural network model proposed by Kennedy and Chua in 1987. After defining the neural dynamics of the proposed neural network, we have shown the existence of an equilibrium point and the asymptotic stability of the equilibrium point of the neural dynamics. As the penalty parameter approaches infinity, an optimal solution of the convex programming problem is shown to be the equilibrium point of the neural dynamics, and each equilibrium point is optimal to the problem.

The configuration of the proposed neural network is described and demonstrated for solving linear programming problems. We have also demonstrated the operational characteristics of the proposed neural network via numerical examples. The results of computer simulations indicate that the proposed neural network is an efficient technique. The introduction of a penalty parameter eliminates neurons and connection weights for Lagrange multipliers which are commonly used in conventional optimization methods, and hence, reduces the size of the neural network. The quality of solutions computed depends mainly on the penalty parameter, while the speed of convergence to the equilibrium point depends on the learning coefficient and the penalty function.

Future potential research directions are listed below.



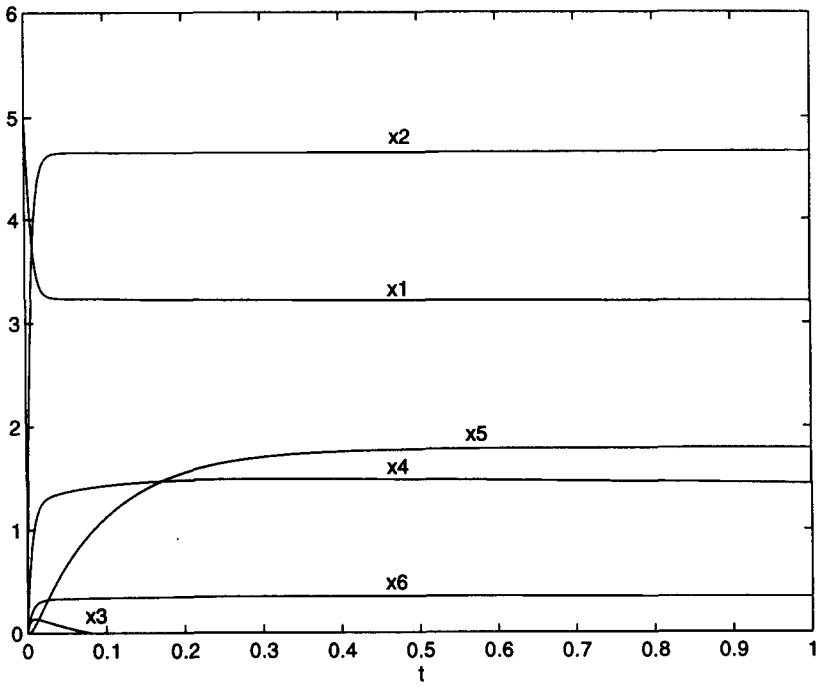
(a) $1/2(g_i(x(t)))^2$.



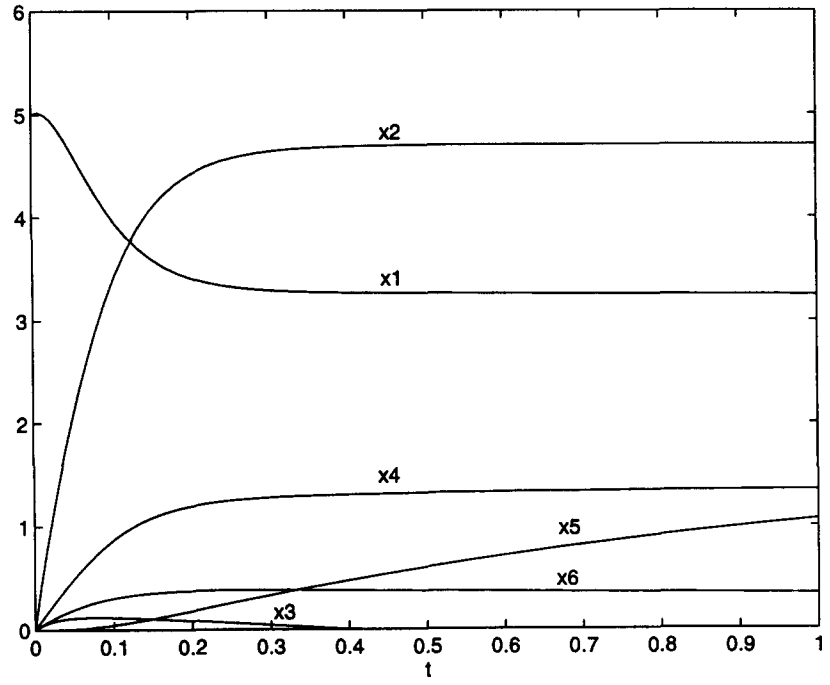
(b) $10 \ln \cosh g_i(x(t))/10$.

Figure 13. State trajectories of Problem (P_1) for different penalty functions with $x(0) = x^3$, $t \in [0, 10]$.

(1) The major improvement of the proposed neural network will be on the quality of the computed solution. We can consider a time-dependent penalty (see [12,27–30]) in the neural network. Such penalty can be an increasing sequence of penalties such that the penalty becomes infinitely large, when the time approaches infinity. Then, the neural dynamics of the proposed neural network with a time-dependent penalty becomes a nonautonomous



(a) $1/2(g_i(x(t)))^2$.



(b) $10 \ln \cosh g_i(x(t))/10$.

Figure 14. State trajectories of Problem (P_1) for different penalty functions with $x(0) = x^3$, $t \in [0, 1]$.

dynamic system instead of an autonomous one. Moreover, the neural dynamics does not have a corresponding autonomous system by which the neural dynamics is bounded above. Therefore, the Liapùnov theory is no longer applicable for analyzing the neural dynamics. A possible direction for analyzing the system is to use other optimization theory for nonlinear programming problems. The Lagrange method has already been well investi-

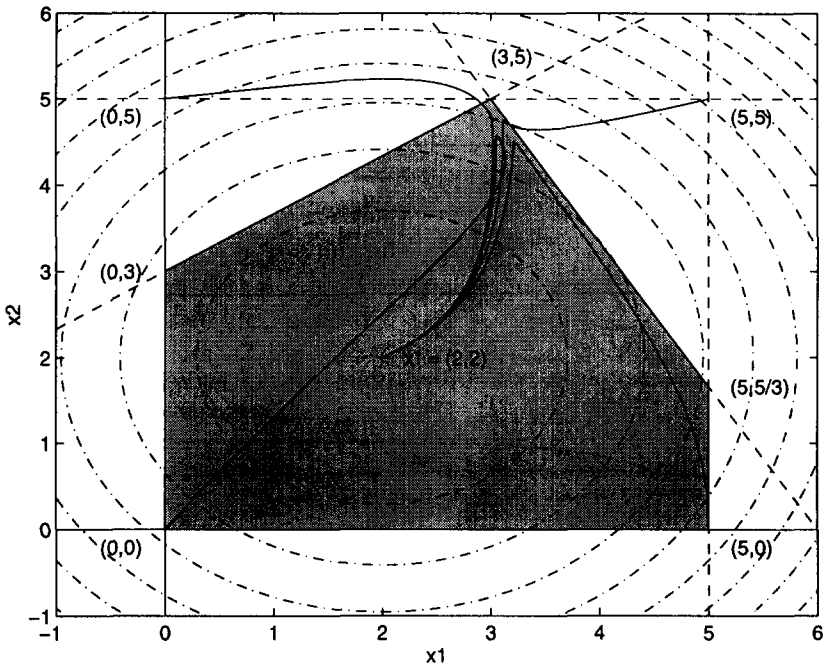


Figure 15. State trajectories of the neural network for Problem (P_2).

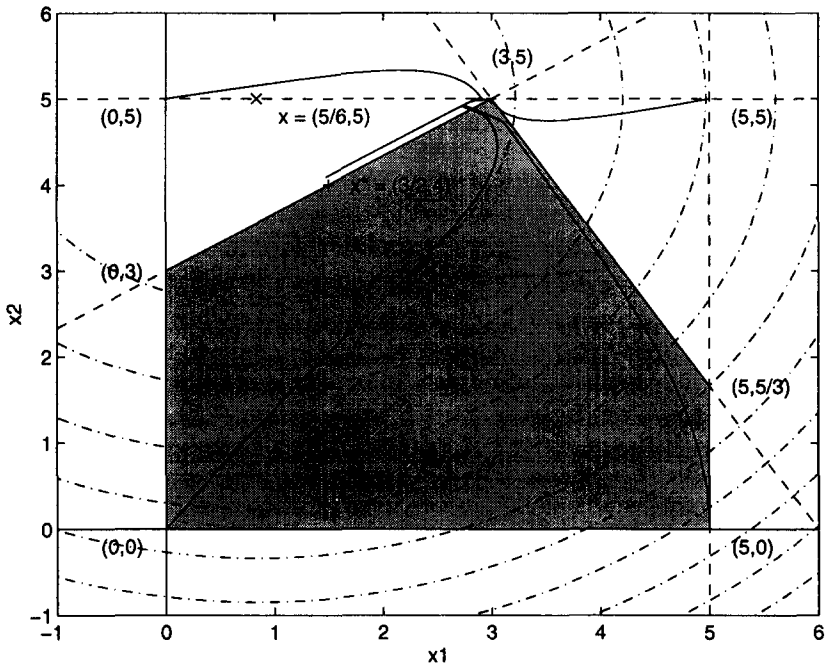


Figure 16. State trajectories of the neural network for Problem (P_3).

gated (see [20,21]). However, the interior point method and barrier method [31] may be applicable for the design of new neural networks.

(2) For the penalty function in the proposed neural dynamics, it is assumed to be continuously differentiable. However, for some convex and piecewise differentiable penalty functions, e.g., absolute penalty functions and Huber's penalty functions, they sometimes can maintain significant speed of convergence by having a constant derivative for points close to the equilibrium points of the neural dynamics. Because these functions are only piece-

wise differentiable, the Liapùnov theory is not applicable, although the corresponding neural energy functions are convex or strictly convex. Therefore, nonsmooth convex analysis (see [32,33]) will be helpful to analyze the neural dynamics having such property.

(3) Numerical experiments can be extended to solve more challenging optimization problems such as the large scale problems [31] or semi-infinite linear programming problems (see [34,35]) for business and industry applications.

REFERENCES

1. L.O. Chua and G.-N. Lin, Nonlinear programming without computation, *IEEE Transactions on Circuits and Systems* **CAS-31** (2), 182–188, (1984).
2. J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the U.S.A.* **79**, 2554–2558, (1982).
3. J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, *Proceedings of the National Academy of Sciences of the U.S.A.* **81**, 3088–3092, (1984).
4. J.B. Dennis, *Mathematical Programming and Electrical Networks*, Chapman and Hall, London, (1959).
5. T.E. Stern, *Theory of Nonlinear Networks and Systems*, Addison-Wesley, New York, (1965).
6. J.J. Hopfield and D. Tank, Neural computation of decisions in optimization problems, *Biological Cybernetics* **58**, 63–70, (1985).
7. A.G. Tsirukis, G.V. Reklaitis and M.F. Tenorio, Nonlinear optimization using generalized Hopfield networks, *Neural Computation* **1** (4), 511–521, (1989).
8. B.A. Pearlmutter, Gradient calculations for dynamic recurrent neural networks: A survey, *IEEE Transactions on Neural Networks* **6** (5), 1212–1228, (1995).
9. S.H. Zak, V. Upatising and S. Hui, Solving linear programming problems with neural networks: A comparative study, *IEEE Transactions on Neural Networks* **6** (1), 96–104, (1995).
10. C.-Y. Maa and M.A. Shanblatt, Linear and quadratic programming neural network analysis, *IEEE Transactions on Neural Networks* **3** (4), 580–594, (1992).
11. C.-Y. Maa and M.A. Shanblatt, A two-phase optimization neural network, *IEEE Transactions on Neural Networks* **3** (6), 1003–1010, (1992).
12. J. Wang, Deterministic annealing neural network for convex programming, *Neural Networks* **7** (4), 629–641, (1994).
13. M.P. Kennedy and L.-O. Chua, Unifying the Tank and Hopfield linear programming network and the Canonical nonlinear programming circuit of Chua and Lin, *IEEE Transactions on Circuits and Systems* **CAS-34**, 210–214, (1987).
14. L.O. Chua and N.N. Wang, Complete stability of autonomous reciprocal nonlinear networks, *Circuit Theory and Applications* **6**, 211–241, (1978).
15. A. Cichocki, R. Unbehauen, K. Weinzierl and R. Hölzel, A new neural network for solving linear programming problems, *European Journal of Operational Research* **93**, 244–256, (1996).
16. M.P. Kennedy and L.-O. Chua, Neural networks for nonlinear programming, *IEEE Transactions on Circuits and Systems* **35**, 554–562, (1988).
17. C.-Y. Maa, C. Chiu and M.A. Shanblatt, A constrained optimization neural net technique for economic power dispatch, *1990 IEEE International Symposium on Circuits and Systems* **4**, 2946–2950, (1990).
18. W.E. Lillo, S. Hui and S.H. Zak, Neural networks for constrained optimization problems, *International Journal of Circuit Theory and Applications* **21** (4), 385–399, (1993).
19. W.E. Lillo, M.H. Loh and S. Hui, On solving constrained optimization problems with neural networks: A penalty method approach, *IEEE Transactions on Neural Networks* **4** (6), 931–940, (1993).
20. D.G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, MA, (1973).
21. M.S. Bazaraa and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley & Sons, New York, (1990).
22. S. Barnett and R.G. Cameron, *Introduction to Mathematical Control Theory*, Second edition, Oxford University Press, New York, (1992).
23. A.M. Liapùnov, Problème gènèral de la stabilitè du mouvement (Reproduction of the French translation in 1907 of a Russian memoire dated 1892), In *Annals of Mathematics Studies*, Volume 17, Princeton University Press, Princeton, NJ, (1947).
24. J. Cronin, *Differential Equations: Introduction and Qualitative Theory*, Second edition, revised and expanded, Marcel Dekker, New York, (1994).
25. S. Haykin, *Neural Networks, A Comprehensive Foundation*, Macmillan, New York, (1994).
26. G.E. Forsythe, M.A. Malcolm and C.B. Moler, *Computation Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, (1977).
27. J. Wang, On the asymptotic properties of recurrent neural networks for optimization, *International Journal of Pattern Recognition and Artificial Intelligence* **5** (4), 581–601, (1991).
28. J. Wang, A time-varying recurrent neural system for convex programming, In *1991 International Joint Conference on Neural Networks (IJCNN '91,* Seattle*)*, pp. 147–152, (1991).

29. J. Wang, Analogue neural network for solving the assignment problem, *Electronics Letters* **28** (11), 1047–1050, (1992).
30. J. Wang, Analysis and design of a recurrent neural network for linear programming, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* **40** (9), 613–618, (1993).
31. S.-C. Fang and S. Puthenpura, *Linear Optimization and Extensions: Theory and Algorithms*, Prentice-Hall, Englewood Cliffs, NJ, (1993).
32. R.T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, (1970).
33. F.H. Clarke, *Optimization and Nonsmooth Analysis*, John Wiley & Sons, New York, (1983).
34. S.-C. Fang and S.Y. Wu, An inexact approach to solving linear semi-infinite programming problems, *Optimization* **28**, 291–299, (1994).
35. S.-C. Fang, C.J. Lin and S.Y. Wu, An unconstrained convex programming approach to linear semi-infinite programming, *SIAM Journal on Optimization* **8** (2), (May 1998).