第 29 卷 第 2 期 自 动 化 学 报 Vol. 29, No. 2
2003 年 3 月 ACTA AUTOMATICA SINICA Mar., 2003

Duality and Neural Networks for Solving Linear Programming[1)]

TIAN Da-Gang
(*College of Management, University of Shanghai Science and Technology, Shanghai* 200093)
(E-mail: ly008369@online.sh.cn)

Abstract By means of new theorems on duality, a sort of recurrent neural network for solving linear programming problems is given, which can be realized easily by circuits. The algorithm's exponentially asymptotic stability in the whole is proven. It makes the neural computing approach for linear programming trend to perfect.

Key words Duality, neural network, linear programming

1 Introduction

The most widely used method for solving linear programming(LP) is the simplex method developed by Dantzig in 1947, but it is not a polynomial-time algorithm[1]. The ellipsoid algorithm proposed by Khachiyan showed that LP problems can be solved in polynomial time, but the algorithm itself is not a practically effective algorithm. The Karmarkar algorithm and its variants appear to be more efficient polynomial-time algorithm[2~5], but much work is still needed in terms of both algorithm refinement and software development. On the other hand, the neural computing approach reveals a kind of novel computing paradigm. Neural networks for solving linear programming problems have been rather extensively studied over the years[6~19]. However, in early neural networks for solving LP problems, issues such as the stability and convergence or the feasibility of equilibrium points were not strictly dealt with[6,7,9,10,12,13] or some assumptions that are difficult to check were included [8,14]. By means of duality theory, Chinese scholars developed a kind of neural network for solving linear programming and quadratic programming problems[15~18], which can solve simultaneously the primal and dual problems, and a rigorous theoretical analysis of some fundamental issues such as convergence are given[18]. In [18], the primal and dual problems must be described simultaneously in a gradient system, it makes the dimensions of the system bigger. Meanwhile, the rate that the solutions converge to the equilibrium point has not been discussed . The neural network in [19] is exponentially asymptotically stable in the whole, but the activation functions with exponential makes it very easy to overflow and difficult to be realized by analog circuits.

In this paper, by a new inequality, we offer new duality theorems related to the linear programming problems, and a kind of new neural network for solving linear programming problems is developed. The proposed neural network has exponentially asymptotically stable in the whole and smaller dimensions and is easily realized by analog circuits. The primal and dual problems can be solved simultaneously by the neural network and the vexatious overflow in computing no longer exists. Meanwhile, we think that by the theory developed in this paper, it is possible to get a new polynomial-time algorithm for LP problems.

The paper is organized as follows. In Section 2, the new duality theorems are presented. In Section 3, the ε-optimal solution is discussed. Section 4 proposes the new simple neural network. The exponentially asymptotic stability in the whole is proven in Section 5. In Section 6, some simulation results are shown. In Section 7, by an example we show

1) Supported by the National 863 Plan(2002AA414310)
Received May 9, 2001;in revised form June 6, 2002
收稿日期 2001-05-09; 收修改稿日期 2002-06-06

that it is possible to get a new polynomial-time algorithm. We conclude the paper in Section 8.

2 Duality

Let function $h(x)$ satisfy the following condition H):

H) 1) $h(x)\in C^1(R), h(x)\geqslant 0,\ h'(x)\geqslant 0$.
2) $\mu h(1/\mu)\xrightarrow{\mu\to 0^+}\infty, h'(x)$ increases monotonically.

Lemma 1. For any $a\in R$ and for any $x\geqslant 0$, the following inequality holds.

$$ax-h(a)\leqslant xh'^{-1}(x)-h(h'^{-1}(x)) \tag{1}$$

Inequality (1) becomes an equality, if and only if $a=h'^{-1}(x)$, where $h'(x)$ denotes the derivative of $h(x)$, and $h'^{-1}(x)$ denotes the inverse function of $h'(x)$.

Proof. Let $f(a)=xa-h(a)$. Evidently, $f(a)$ attains its maximum at $a=h'^{-1}(x)$. That is, $ax-h(a)\leqslant xh'^{-1}(x)-h(h'^{-1}(x))$ □

Consider a linear programming problem as follows.

Program P: Min $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$
s. t. $A\boldsymbol{x}=\boldsymbol{b},\ \boldsymbol{x}\geqslant\boldsymbol{0}$

The linear dual of Program $\boldsymbol{P}$ is given as follows.

Program Q: Max $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{w}$
s. t. $A^{\mathrm{T}}\boldsymbol{w}\leqslant\boldsymbol{c}$
$\boldsymbol{w}\in R^m$

where $\boldsymbol{c}$ and $\boldsymbol{x}$ are n-dimensional column vectors, $\boldsymbol{b}$ and $\boldsymbol{w}$ are m-dimensional column vectors, and A is an $m\times n$ $(m<n)$ matrix.

Instead of solving Program P directly, for $\mu>0$, we consider the nonlinear programs as follows

Program $\boldsymbol{P}_\mu$: Min $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}+\mu\sum_{j=1}^{n}x_jh'^{-1}(x_j)-\mu\sum_{j=1}^{n}h(h'^{-1}(x_j))$
s. t. $A\boldsymbol{x}=\boldsymbol{b},\ \boldsymbol{x}\geqslant\boldsymbol{0}$

We shall show that Program $\boldsymbol{P}_\mu$ has a dual program as follows:

Program $\boldsymbol{D}_\mu$: Max $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{w}-\mu\sum_{j=1}^{n}h((\sum_{i=1}^{m}a_{ij}w_i-c_j)/\mu)$
s. t. $\boldsymbol{w}\in R^m$

Theorem 1(Weak duality). Min$(\boldsymbol{P}_\mu)\geqslant$Max$(\boldsymbol{D}_\mu)$.

Proof. For any $\boldsymbol{w}\in R^m$ and $j=1,2,\cdots,n$, let $a_j=(\sum_{i=1}^{m}a_{ij}w_i-c_j)/\mu$. Suppose $x_j\geqslant 0$ satisfies $\sum_{j=1}^{n}a_{ij}x_j=b_i$, for $i=1,2,\cdots,m$. Then by (1),

$$x_ja_j-h(a_j)\leqslant x_jh'^{-1}(x_j)-h(h'^{-1}(x_j)),\ \text{for } j=1,2,\cdots,n \tag{2}$$

By summing over j, we have

$$\boldsymbol{b}^{\mathrm{T}}\boldsymbol{w}-\mu\sum_{j=1}^{n}h\Big(\Big(\sum_{i=1}^{m}a_{ij}w_i-c_j\Big)\Big/\mu\Big)\leqslant\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}+\mu\sum_{j=1}^{n}x_jh'^{-1}(x_j)-\mu\sum_{j=1}^{n}h(h'^{-1}(x_j)) \tag{3}$$

That is, Min$(\boldsymbol{P}_\mu)\geqslant$Max$(\boldsymbol{D}_\mu)$. □

Remark. Let $a_j=(\sum a_{ij}w_i-c_j)/\mu-1$ and $h(x)=\mathrm{e}^x$. Then inequality (3) becomes (6) of [22]. For $y<0$, let $h(y)=-\ln(-y)$. It is easy to get the original barrier function $\mu\sum_{j=1}^{n}\ln x_j$ and its duality.

Theorem 2. Given that $\boldsymbol{w}(\mu)\in R^m$ and $\boldsymbol{x}(\mu)\in R^n$ such that $A\boldsymbol{x}(\mu)=\boldsymbol{b}$ and $\boldsymbol{x}(\mu)\geqslant\boldsymbol{0}$ if

$$x_j(\mu)=h'\Big(\Big(\sum_{i=1}^{m}a_{ij}w_i(\mu)-c_j\Big)\Big/\mu\Big),\ \text{for } j=1,2,\cdots,n \tag{4}$$

then $\boldsymbol{w}(\mu)$ is an optimal solution to Program $\boldsymbol{D}_\mu$ and $\boldsymbol{x}(\mu)$ is an optimal solution to Program $\boldsymbol{P}_\mu$. Moreover, Min$(\boldsymbol{P}_\mu)=$Max$(\boldsymbol{D}_\mu)$.

Proof. By lemma, inequality (1) becomes an equality if and only if $a = h'^{-1}(x)$. Hence inequality (3) becomes an equality if and only if

$$h'(a_j) = h'\Big(\Big(\sum_{i=1}^{m} a_{ij} w_i(\mu) - c_j\Big)\Big/\mu\Big) = x_j. \qquad \square$$

Let $f(\boldsymbol{w},\mu) = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{w} - \mu \sum_{j=1}^{n} h\Big(\big(\sum_{i=1}^{m} a_{ij} w_i - c_j\big)\Big/\mu\Big)$.

Theorem 3(Strong duality). If the set of optimal solutions of Program $\boldsymbol{D}$ is non-empty and bounded, then there exists $\mu_0 > 0$ such that Program $\boldsymbol{D}_\mu$ and Program $\boldsymbol{P}_\mu$ all have optimal solutions as $\mu \in (0, \mu_0)$. Let $\boldsymbol{w}(\mu) \in R^m$ be an optimal solution to $\boldsymbol{D}_\mu$. Then formula (4) provides a dual-to-primal conversion, which defines the optimal solution $\boldsymbol{x}(\mu)$ to Program $\boldsymbol{P}_\mu$. Moreover, $\mathrm{Min}(\boldsymbol{P}_\mu) = \mathrm{Max}(\boldsymbol{D}_\mu)$.

Proof. By Proposition 1 of [20], there exists $\mu_0 > 0$ such that $f(\boldsymbol{w},\mu)$ can attain a maximum as $\mu \in (0,\mu_0)$. Let $\boldsymbol{w}(\mu)$ be an optimal solution to Program $\boldsymbol{D}_\mu$. If $\nabla f(\boldsymbol{w},\mu)$ denotes the gradient of $f(\boldsymbol{w},\mu)$, then $\nabla f(\boldsymbol{w}(\mu),\mu) = \boldsymbol{0}$. That is,

$$b_i = \sum_{j=1}^{n} a_{ij} h'\Big(\Big(\sum_{k=1}^{m} a_{kj} w_k(\mu) - c_j\Big)\Big/\mu\Big) = \sum_{j=1}^{n} a_{ij} x_j(\mu), \text{ for } i = 1,2,\cdots,m$$

Combining the above equality with Theorem 2 gives Theorem 3. $\square$

3 An ε-optimal solution

By Theorem 3 and using Proposition 1, Proposition 2 and (2.1) of [20], we have Theorem 4 as follows.

Theorem 4. If the set of the optimal solutions of Program $\boldsymbol{D}$ is non-empty and bounded, then there exists $\mu_0 > 0$ and a compact set K such that $\boldsymbol{w}(\mu) \in K$ as $\mu \in (0,\mu_0)$, where $\boldsymbol{w}(\mu)$ denotes the optimal solution to Program $\boldsymbol{D}_\mu$. If $\lim_{\mu_k \to 0^+} \boldsymbol{w}(\mu_k) = \boldsymbol{w}^*$, and $\boldsymbol{x}(\mu_k)$ is defined by (4), then $\boldsymbol{w}^*$ is an optimal solution to Program $\boldsymbol{D}$ and $\boldsymbol{x}(\mu_k)$ approaches to an optimal solution of Program $\boldsymbol{P}$ as $\mu_k \to 0^+$.

Theorem 5. Suppose that $\boldsymbol{x}(\mu)$ solves Program $\boldsymbol{P}_\mu$. Then $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu)$ decreases monotonically, as $\mu > 0$ goes to 0. Moreover, for $\mu_1 > \mu_2 > 0$, if $\boldsymbol{x}(\mu_1)$ and $\boldsymbol{x}(\mu_2)$ solve Programs $\boldsymbol{P}_{\mu_1}$ and $\boldsymbol{P}_{\mu_2}$, respectively, then

$$0 \leqslant \mu_2[p(\boldsymbol{x}(\mu_2)) - p(\boldsymbol{x}(\mu_1))] \leqslant \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu_1) - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu_2) \leqslant \mu_1[p(\boldsymbol{x}(\mu_2)) - p(\boldsymbol{x}(\mu_1))] \quad (5)$$

where $p(\boldsymbol{x}) = \sum_{j=1}^{n} x_j h'^{-1}(x_j) - \sum_{j=1}^{n} h(h'^{-1}(x_j))$.

Proof. Since $\boldsymbol{x}(\mu_1)$ and $\boldsymbol{x}(\mu_2)$ solve Programs $\boldsymbol{P}_{\mu_1}$ and $\boldsymbol{P}_{\mu_2}$, respectively, we have

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu_1) + \mu_1 p(\boldsymbol{x}(\mu_1)) \leqslant \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu_2) + \mu_1 p(\boldsymbol{x}(\mu_2)) \quad (6)$$

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu_2) + \mu_2 p(\boldsymbol{x}(\mu_2)) \leqslant \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu_1) + \mu_2 p(\boldsymbol{x}(\mu_1)) \quad (7)$$

Multiplying (7) by -1 and adding the resultant inequality to (6), we obtain

$$0 \leqslant (\mu_1 - \mu_2)[p(\boldsymbol{x}(\mu_2)) - p(\boldsymbol{x}(\mu_1))]$$

Since $\mu_1 > \mu_2$, we know that

$$p(\boldsymbol{x}(\mu_2)) \geqslant p(\boldsymbol{x}(\mu_1)) \quad (8)$$

After rearranging terms in (6) and (7) and using (8), we further have

$$0 \leqslant \mu_2[p(\boldsymbol{x}(\mu_2)) - p(\boldsymbol{x}(\mu_1))] \leqslant \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu_1) - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu_2) \leqslant \mu_1[p(\boldsymbol{x}(\mu_2)) - p(\boldsymbol{x}(\mu_1))] \quad \square$$

Theorem 6. Suppose $h'^{-1}(\boldsymbol{x})$ has left-hand derivative and right-hand derivative, and that $\boldsymbol{x}^*$ is an optimal solution to Program $\boldsymbol{P}$. Then $p(\boldsymbol{x}^*) \geqslant p(\boldsymbol{x}(\mu))$. Let $L = \max\{|p(\boldsymbol{x}^*)|, h(0)\}$. Then $0 \leqslant \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu) - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^* \leqslant \mu[p(\boldsymbol{x}^*) - p(\boldsymbol{x}(\mu))] \leqslant 2\mu L$

Therefore, as $\mu < \frac{\varepsilon}{2L}$, $\boldsymbol{x}(\mu)$ is an ε-optimal solution to Program $\boldsymbol{P}$.

Proof. Since $\boldsymbol{x}(\mu)$ is feasible, $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu) \geqslant \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^*$, and

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(\mu) + \mu p(\boldsymbol{x}(\mu)) \leqslant \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^* + \mu p(\boldsymbol{x}^*)$$

Therefore,

$$0 \leqslant c^{T}x(\mu) - c^{T}x^{*} \leqslant \mu[p(x^{*}) - p(x(\mu))]$$

It is easy to know that when $h'^{-1}(x)$ has left-hand derivative and right-hand derivative, $p(x)$ is differentiable and $p'(x) = h'^{-1}(x)$. So $p(x)$ attains its minimum $-h(0)$ at $h'(0)$ and

$$\mu[p(x^{*}) - p(x(\mu))] \leqslant 2\max\{|p(x^{*})|, |p(x(\mu))|\} \leqslant 2\max\{|p(x^{*})|, h(0)\} \square$$

4 A simpler perturbation function and the relative neural network

For $\beta > 1$, we define $h_1(x) = \begin{cases} \frac{1}{2}x^2 + \frac{1}{\beta}x + \frac{1}{2}, & x \geqslant 0, \\ \frac{1}{2\beta^2}(x+\beta)^2, & -\frac{\beta}{2} < x < 0, \\ \frac{1}{8}e^{\frac{4}{\beta}x+2}, & x \leqslant -\frac{\beta}{2}. \end{cases}$

It is obvious that $h_1(x)$ satisfies condition H). Consider the following nonlinear program.

Program f_1: $\quad \operatorname{Max} b^{T}w - \mu\sum_{j=1}^{n} h_1\left((\sum_{i=1}^{m} a_{ij}w_i - c_j)/\mu\right)$

Suppose that the set of the optimal solutions of Program $\boldsymbol{D}$ is non-empty and bounded. By Proposition 1 and its proof in [20], there are $\mu_0 > 0$ and compact set K, when $\mu \in (0, \mu_0)$, for all $\beta > 1$, the optimal solution of Program f_1 belongs to K. Therefore, for β big enough, the optimal solution of Program f_1 is identical to the optimal solution of Program $\boldsymbol{F}$ as follows

Program F. $\operatorname{Max} F(w, \mu) = b^{T}w - \mu\sum_{j=1}^{n} H\left((\sum_{i=1}^{m} a_{ij}w_i - c_j)/\mu\right)$, where

$$H(x) = \begin{cases} \frac{1}{2}x^2 + \frac{1}{\beta}x + \frac{1}{2}, & x \geqslant 0, \\ \frac{1}{2\beta^2}(x+\beta)^2, & x < 0. \end{cases} \tag{9}$$

There is no longer an exponential function part in (9).

The derivative of $H(x)$ is $H'(x) = \begin{cases} x + \frac{1}{\beta}, x \geqslant 0, \\ \frac{1}{\beta^2}x + \frac{1}{\beta}, x < 0. \end{cases}$

$H'(x)$ is a piecewise linear function. The gradient system of Program F is as follows:

$$\frac{dw_i}{dt} = b_i - \sum_{j=1}^{n} a_{ij} H'\left(\frac{\sum_{k=1}^{m} a_{kj}w_k - c_j}{\mu}\right), \quad i = 1, 2, \cdots, m \tag{10}$$

A proposed recurrent neural network for solving (10) consists of n massively connected artificial neurons, m integrators, $m+n$ adders as shown in Fig. 1 where $r_i(w) = \sum_{i=1}^{m} a_{ij}w_i - c_i$, $o_i = r_i(w)/\mu$, $i = 1, 2, \cdots, n$ and n neurons are corresponding to n elements of activation states $x(t)$.

Equation (4) makes the primal and dual problems be solved simultaneously by the proposed neural networks. We only solve a system with m dimensions, not $m+n$ dimensions as in [18].

5 The convergence of the proposed neural networks

Theorem 7. Suppose that rank $(A) = m$ and the set of the optimal solutions of Program $\boldsymbol{D}$ is non-empty and bounded. Then the equilibrium point is exponentially asymptotically stable in the whole.

Proof. It is easy to know that for any x_1 and x_2, there are $\alpha(x_1, x_2)$, and $\frac{1}{\beta^2} \leqslant \alpha(x_1, x_2)$

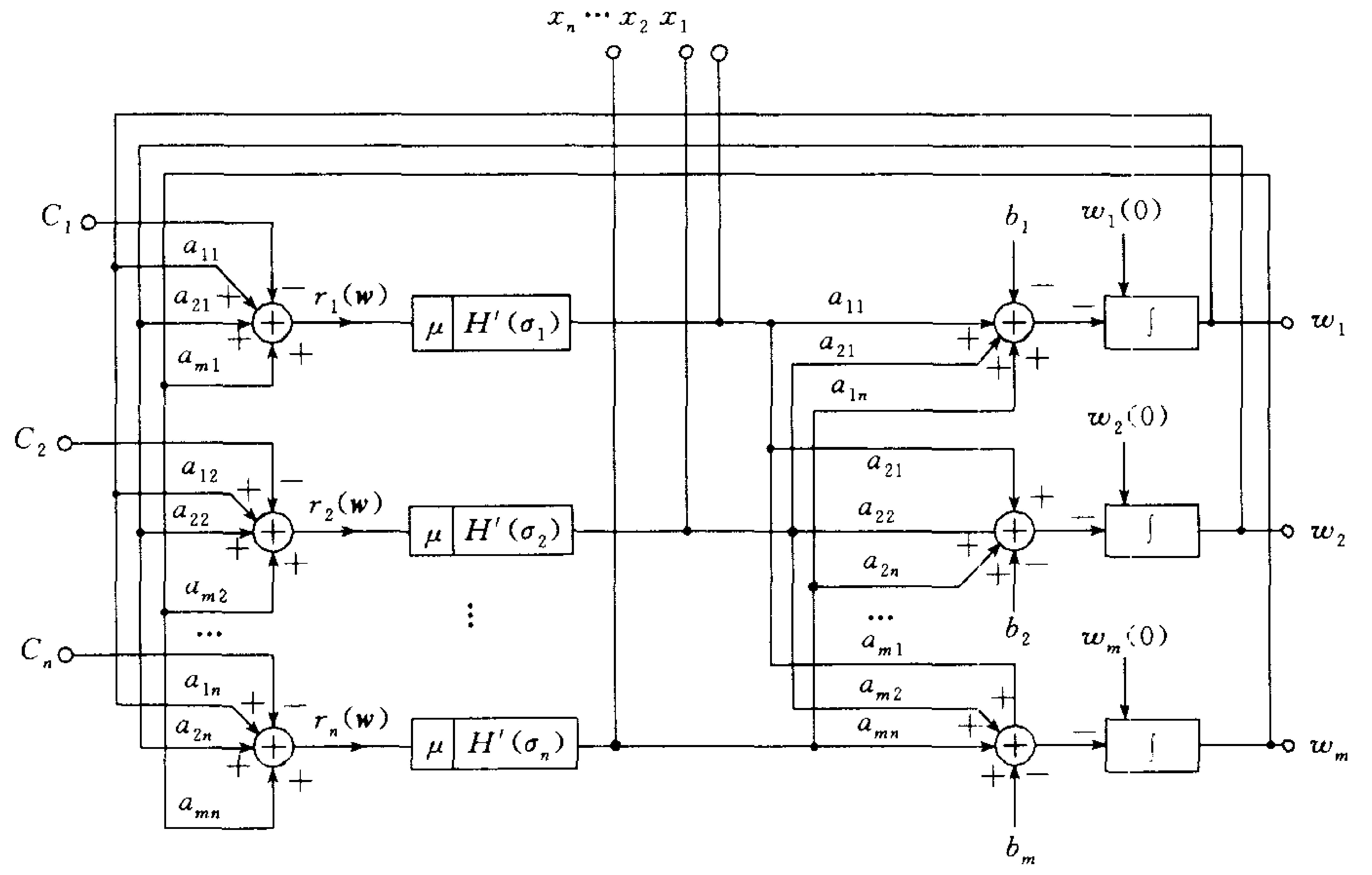

Fig. 1 Architecture of the proposed recurrent neural networks for (10)

$\leqslant 1$, such that

$$H'(x_1) - H'(x_2) = \alpha(x_1, x_2)(x_1 - x_2) \tag{11}$$

Let $w(\mu)$ be an optimal solution of Program $\boldsymbol{F}$, and let $v = w - w(\mu)$. Then the gradient system of Program $\boldsymbol{F}$ becomes as follows:

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{\mathrm{d}w}{\mathrm{d}t} = \nabla F(v + w(\mu), \mu) = \nabla F(v + w(\mu), \mu) - \nabla F(w(\mu), \mu) =$$
$$A\left[H'\left(\frac{A^{\mathrm{T}}w - c}{\mu}\right) - H'\left(\frac{A^{\mathrm{T}}(w + v) - c}{\mu}\right)\right] = -A\Lambda(x_1, x_2)A^{\mathrm{T}}v/\mu .$$

We get $\frac{1}{2}\frac{\mathrm{d}\|v\|^2}{\mathrm{d}t} = -v^{\mathrm{T}}A\Lambda(x_1, x_2)A^{\mathrm{T}}v/\mu \leqslant -\frac{1}{\mu\beta^2}v^{\mathrm{T}}AA^{\mathrm{T}}v \leqslant -\frac{\lambda_m}{\mu\beta^2}\|v\|^2$,

where $\Lambda(x_1, x_2) = \mathrm{diag}(\alpha_1(x_1, x_2), \alpha_2(x_1, x_2), \cdots, \alpha_n(x_1, x_2))$, $\alpha_i(x_1, x_2)$ is as in (11), and λ_m is the minimum eigenvalues of matrix AA^{T}. Thus $\|v\| \leqslant \|v(0)\| \mathrm{e}^{-\frac{\lambda_m}{\mu\beta^2}t}$. □

6 Simulation examples

The examples come from [12], [14], and [19], respectively. In computing, we used the Runge-Kutta method and the half step rectifying.

Example 1[12]. min $-8x_1 - 8x_2 - 5x_3 - 5x_4$

$$\begin{aligned}
\text{s. t.}\quad & 5x_1 - 5x_2 - x_5 = 0\\
& -2x_1 + 3x_2 - x_6 = 0\\
& x_1 + x_3 = 40\\
& x_2 + x_4 = 60\\
& x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6 \geqslant 0
\end{aligned}$$

Example 2[14]. min $-3x_1 - x_2 - 3x_3$

$$\begin{aligned}
\text{s. t.}\quad & 2x_1 + x_2 + x_3 + x_4 = 2\\
& x_1 + 2x_2 + 3x_3 + x_5 = 5\\
& 2x_1 + 2x_2 + x_3 + x_6 = 6\\
& x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6 \geqslant 0
\end{aligned}$$

Example 3[19].

$$\begin{array}{llllllll}
\min & 3x_1 + 2x_2 & + x_3 + 4x_4 & & & & & \\
\text{s. t.} & 2x_1 + 4x_2 + 5x_3 & + x_4 & - x_5 & & & = 230, \\
& 3x_1 - x_2 + 7x_3 - 2x_4 & & & - x_6 & & = 46, \\
& 5x_1 + 2x_2 & + x_3 + 6x_4 & & & - x_7 & = 345, \\
& x_1, \ x_2, \ x_3, \ x_4, & x_5, & x_6, & x_7 \geqslant 0. & &
\end{array}$$

The parameters are: $\beta=1.0e+10$, μ changes from 0.001 to 0.0000001 in Example 1 and Example 3, from 0.001 to 0.00001 in Example 2, the elements of initial vectors are all 1.

Fig. 2 and 3 illustrate the convergent state trajectories of Examplel. The optimal value is -739.99973 and the optimal solutions are as follows:

$$\boldsymbol{w} = (0.59999714, -0.00000410, -10.99998980, -4.99999790)^T;$$
$$\boldsymbol{x} = (40.00002478, 39.99997311, 0.0, 19.99999956, 0.0, 40.00000207)^T$$

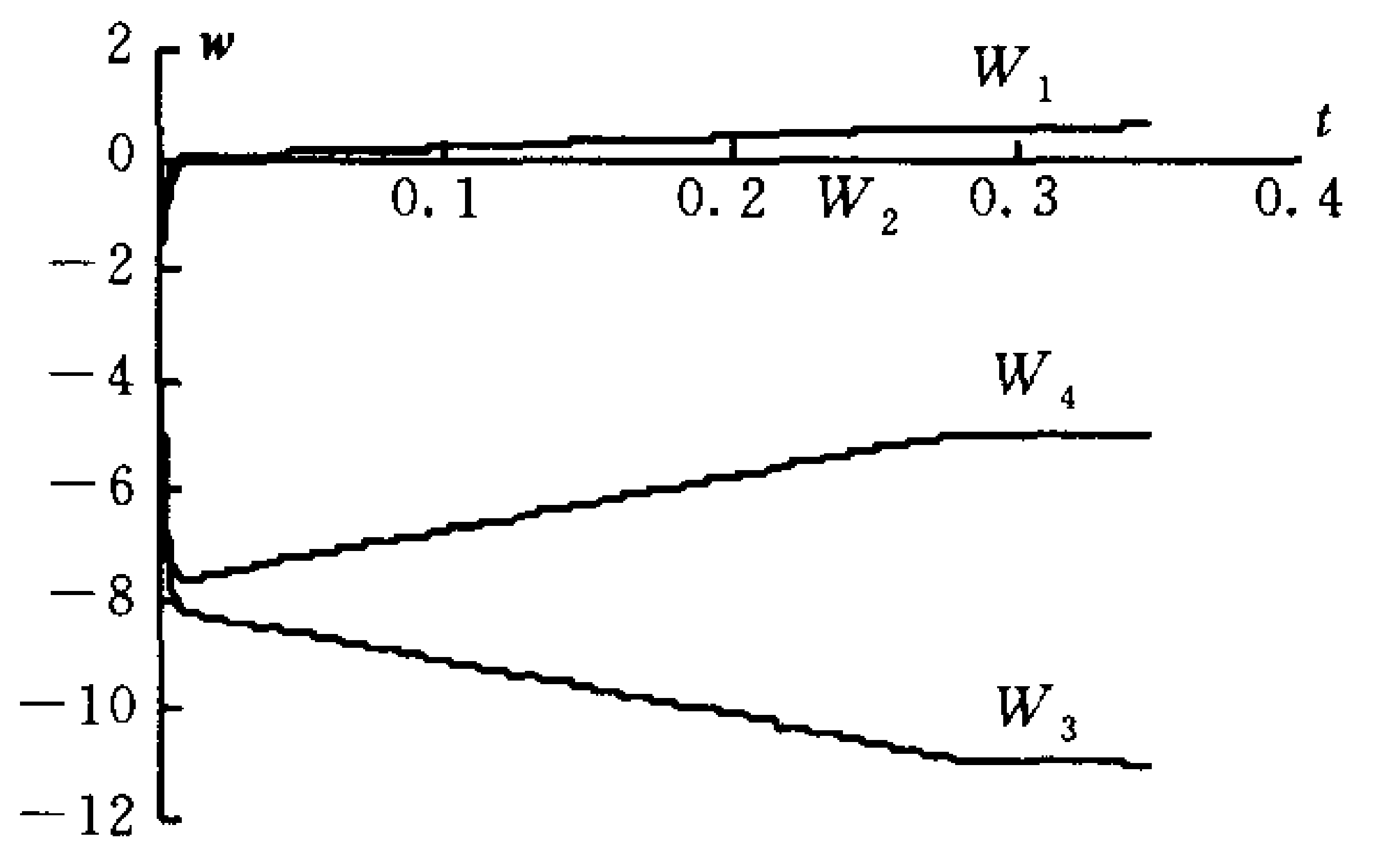

Fig. 2 Convergent state trajectories in Example 1

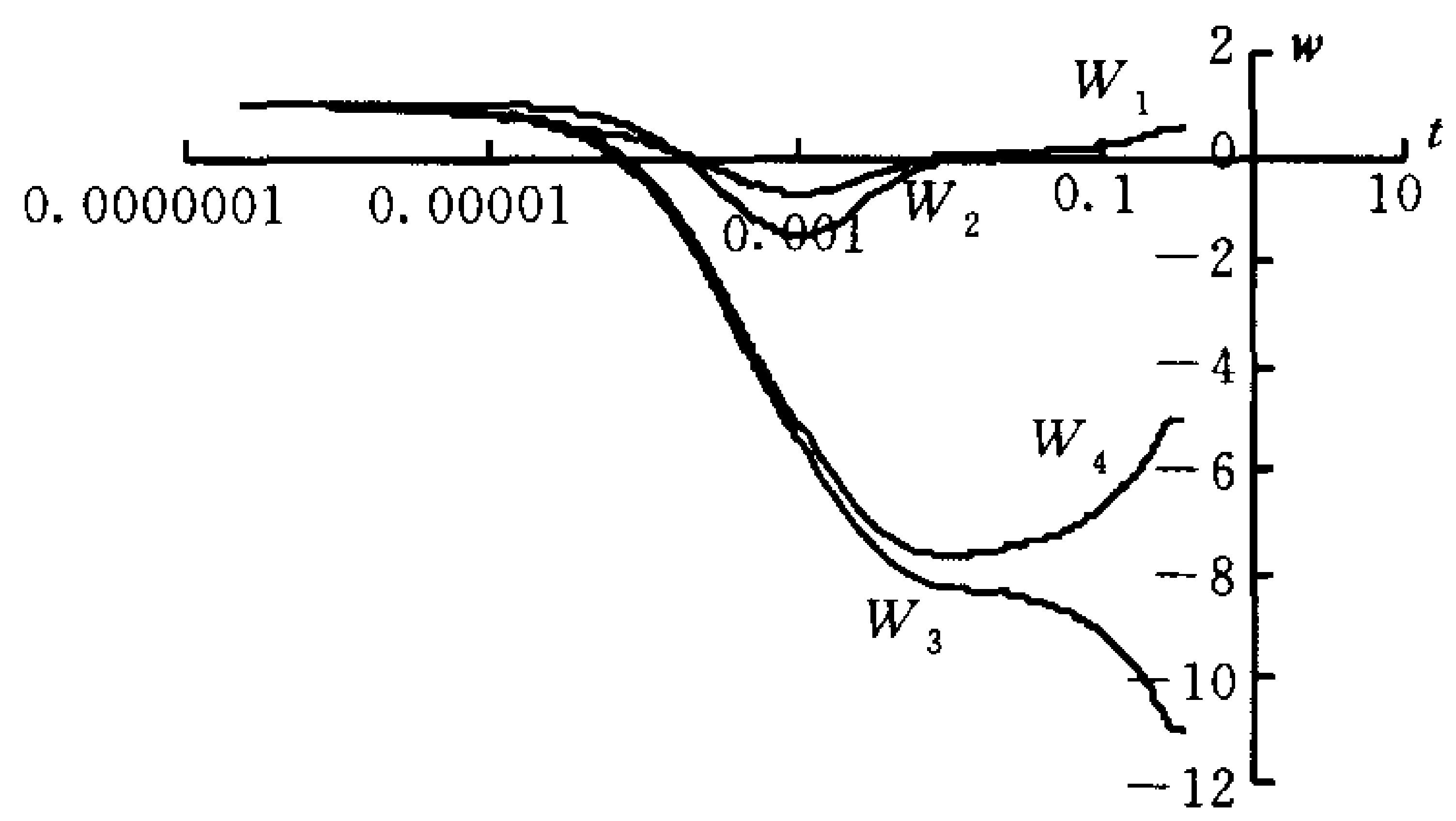

Fig. 3 Convergent state trajectories with logarithm time scales in Example 1

Fig. 4 and 5 illustrate the convergent state trajectories of Example 2. The optimal value is -5.39987900 and the optimal solutions are as follows:

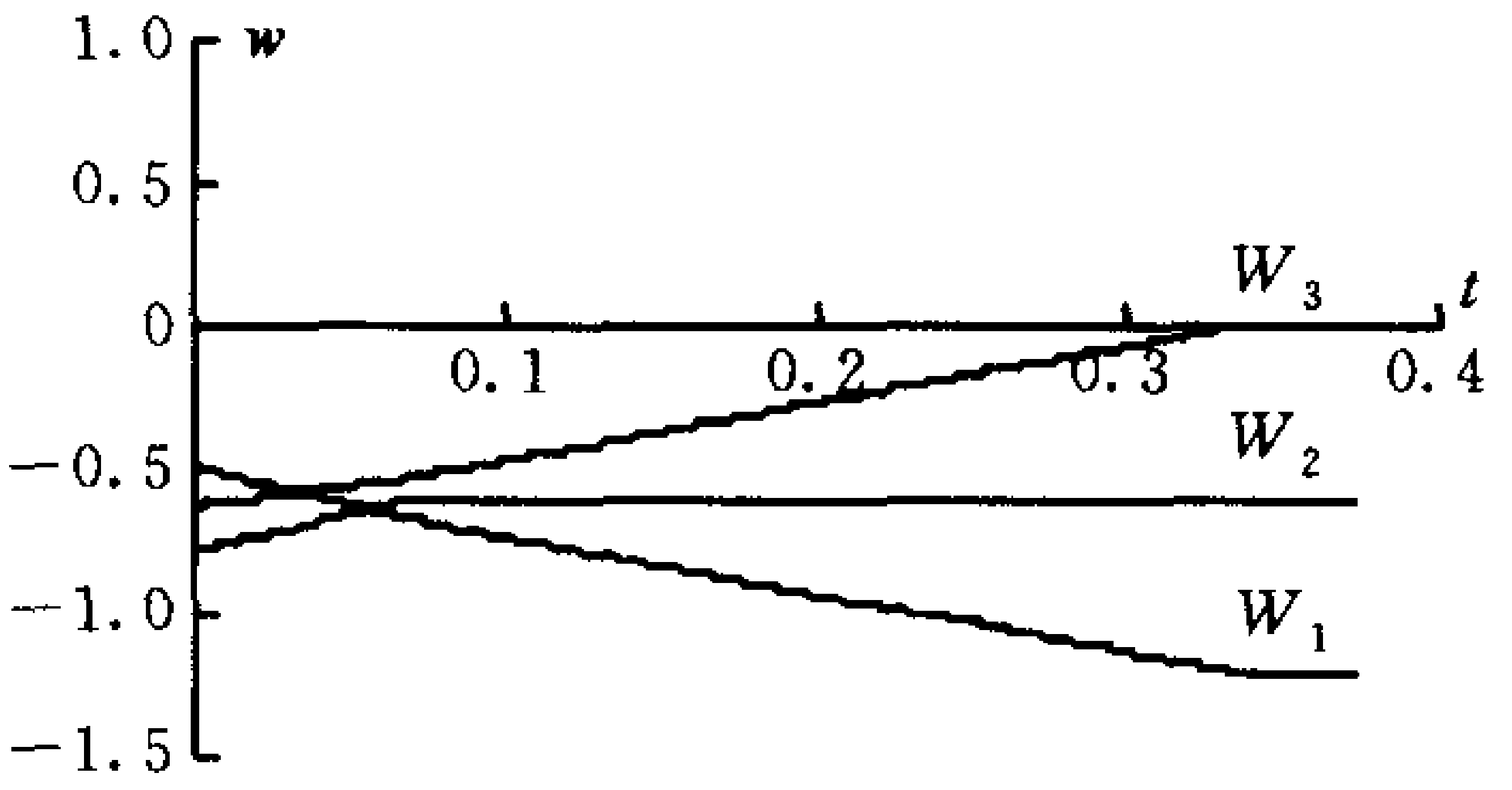

Fig. 4 Convergent state trajectories in Example 2

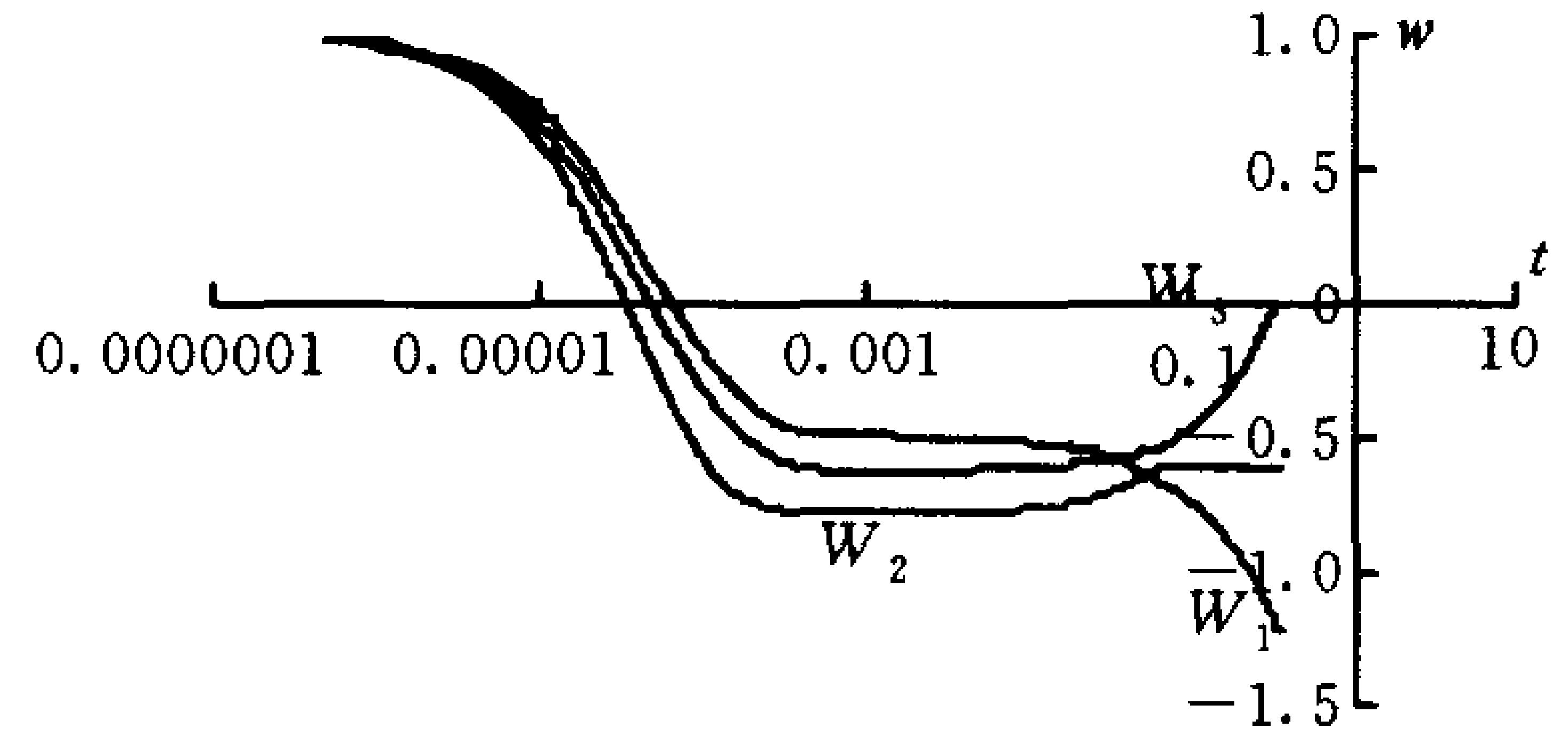

Fig. 5 Convergent state trajectories with logarithm time scales in Example 2

$$\boldsymbol{w} = (-1.20004801, -0.59999200, 0.00005001)^T,$$
$$\boldsymbol{x} = (0.19967117, 0.0, 1.60011964, 0.0, 0.0, 4.00094638)^T.$$

Fig. 6 and 7 illustrate the convergent state trajectories of Example 3. The optimal value is 215.00444435 and the optimal solutions are as follows:

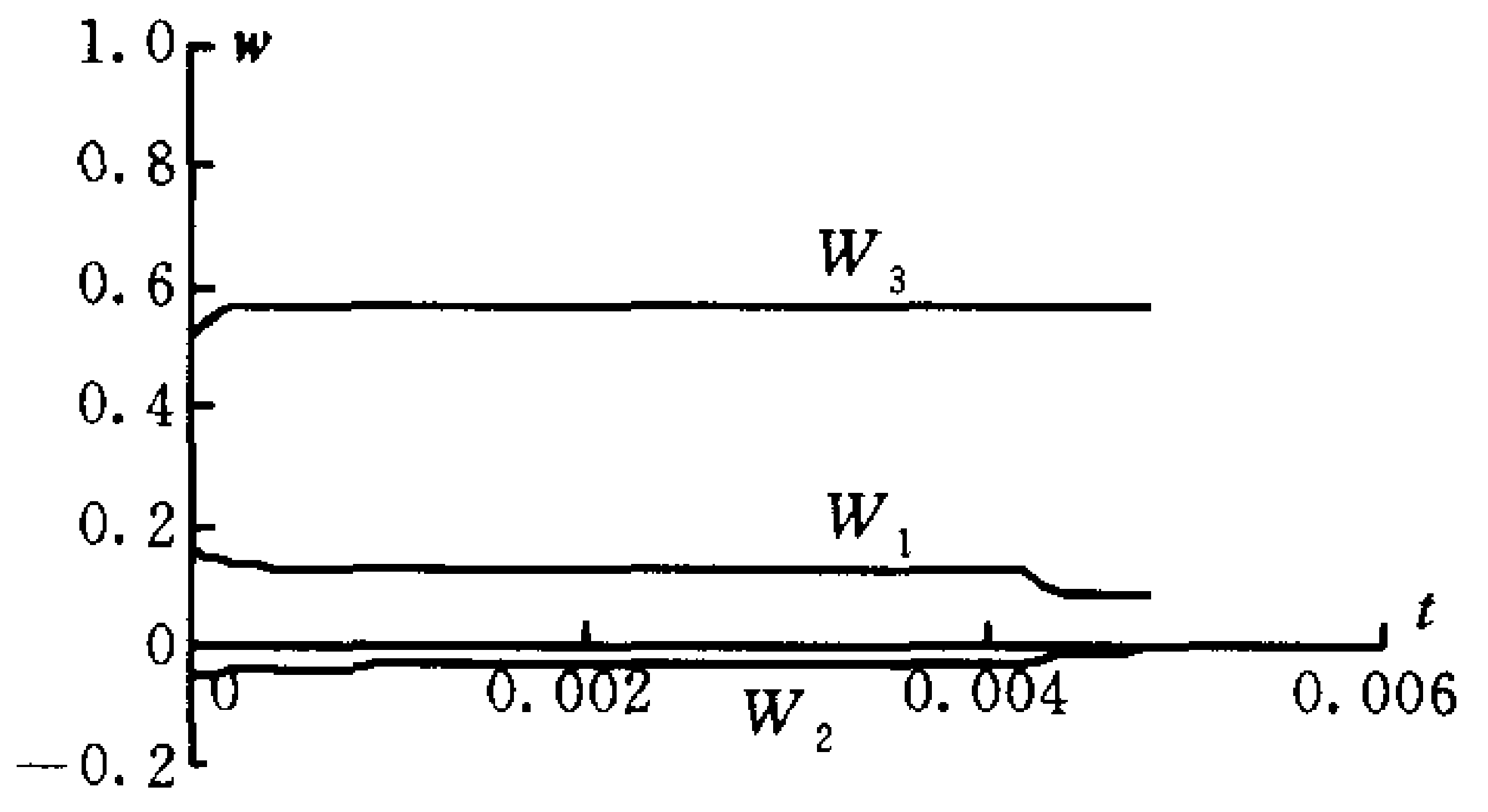

Fig. 6 Convergent state trajectories in Example 3

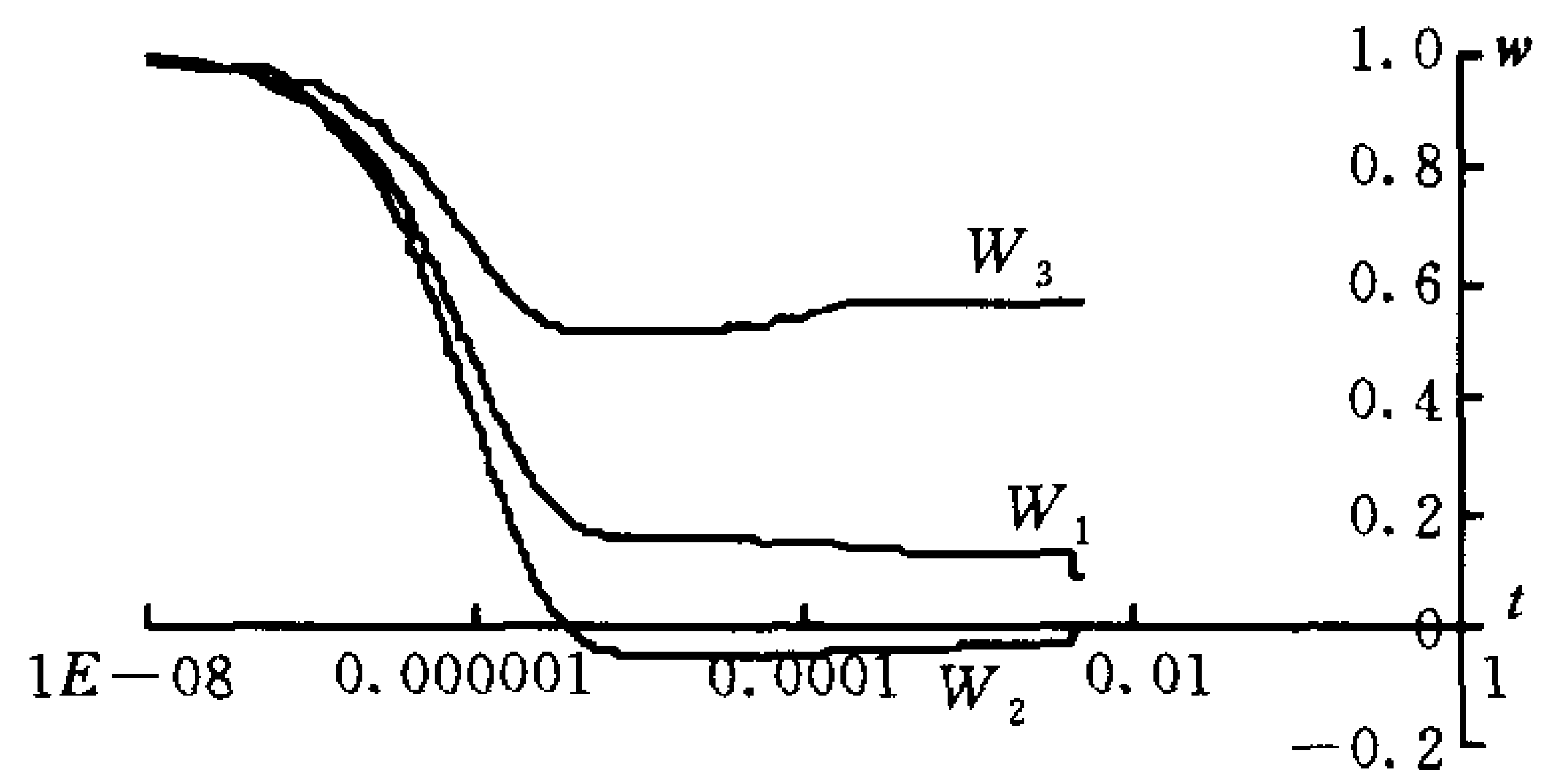

Fig. 7 Convergent state trajectories with logarithm time scales in Example 3

$w = (0.08699704, -0.00002900, 0.56521990)^T$,

$x = (64.99999771, 0.0, 19.99999625, 0.0, 0.0, 289.00000034, 0.0)^T$.

7 About the polynomial-time algorithm

In the discussion above, a kind of perturbation function (see (9)) different from the original logarithm function is used. It makes the algorithm not trend to the interior of feasible region as with the logarithm perturbation. The algorithm with the new perturbation function may be like the simple method slighly, but trend to the interior side of the restriction super-plane. These characteristics make the algorithm search a better point in the whole. We will show this by an example as follows. The example comes from Klee-Schrijver, and we take it from [21].

Example 4(Klee-Schrijver).

$$\begin{array}{lllllll}
\max & 2^{n-1}x_1 + & 2^{n-3}x_3 + \cdots & & + x_{2n-1} & & \\
\text{s. t.} & x_1 + x_2 & & & & & = 5, \\
& 4x_1 + 0 & + x_3 + x_4 & & & & = 5^2, \\
& \vdots \quad \vdots & & & & & \vdots \\
& 2^{n-1}x_1 & + 0 + 2^{n-2}x_3 & + 0 + \cdots + x_{2n-3} + x_{2n-2} & & & = 5^{n-1}, \\
& 2^n x_1 & + 0 + 2^{n-1}x_3 & + 0 + \cdots + 4x_{2n-3} & + 0 + x_{2n-1} + x_{2n} & & = 5^n. \\
& x_1, \ x_2, & \cdots, & & & x_{2n}, & \geqslant 0.
\end{array}$$

Solving Example 4 by the simplex method, one initiates from the feasible base $\boldsymbol{B} = (\boldsymbol{P}_2, \boldsymbol{P}_4, \cdots, \boldsymbol{P}_{2n})$. It has been proved that it must take $2^n - 1$ iterations to attain the optimal solution.

With the help of perturbation function (9), using Newton direction method, and initiating from every vertex of the feasible region, the computing cases for $n = 5 \sim 10$ of Example 4 are shown in Table 1 .

Table 1 The computing cases for $n = 5 \sim 10$ of Example 4 with the proposed method

n	5	6	7	8	9	10
Maximum iterative numbers	13	19	28	42	61	77
Average iterative numbers	8.5	11	14	19	22	25
Parameters	$\mu = 10e-13$, $\beta = 10e+21$					

It is shown that the maximum number of iterations is less than n^2 and the average number of iterations is less than $3n$. The author thinks that work on this way may develop new polynomial-time algorithms which can remain the strongpoint of less average number of iterations in the simplex method.

8 Conclusion

For linear programming problems, using different perturbation functions, will bring about different effects. With the help of new duality theorems proposed in this paper, new methods and new neural networks for solving linear programming problems can be developed. Meanwhile, the means used in the paper is of interest in developing new practicable polynomial-time algorithms for linear programming.

References

1 Klee V, Minty G J. How good is the simplex algorithm? In: Inequalities III, Shisha O ed. New York: Academic Press, 1972. 159～175

2 Bland R G, Goldfarb D, Todd M J. The ellipsoid method: A survey. *Operations Research*, 1981, **29**(6): 1039～1091

3 Karmarkar N K. A new polynomial-time algorithm for linear programming. *Combinatorica*, 1984, **4**(4): 375～395

4 Hooker J N. Karmarkar's linear programming algorithm. *Interfaces*, 1986, **14**(1): 75～90

5 Ghellinck G De, Vial J P. A polynomial Newton method for linear programming. *Algorithmica*, 1986, **1**(3): 425～453
6 Tank D W, Hopfield J J. Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems*, 1986, **33**(4): 533～541
7 Maa C Y, Shanblatt M. Linear and quadratic programming neural networks analysis. *IEEE Transactions on Neural Networks*, 1992, **3**(7): 580～594
8 Kennedy M P, Chua L O. Neural networks for nonlinear programming. *IEEE Transactions on Circuits and Systems*, 1988, **35**(5): 554～562
9 Rodriguez-Vazquez A, Dominguez-Castro R, Huertas J L, Sanchez-Sinencio E. Nonlinear swiched-capacitor 'neural networks' for optimization problems. *IEEE Transactions on Circuits and Systems*, 1990, **37**(3): 384～397
10 Glazos M P, Hui S, Zak S H. Sliding modes in solving convex programming problems. *SIAM Journal of Control and Optimization*, 1998, **36**(2): 680～697
11 Zak S H, Upatising V, Lillo E, Hui S. A dynamical systems approach to solving linear programming problems. In: Differential Equations, Dynamical Systems, and Control Science, Elworthy K D, Everitt W N, Lee E B Eds. New York: Marcel Dekker, 1994
12 Zak H S, Upatising V, Hui S. Solving LP problem with neural networks: A comparative study. *IEEE Transactions on Neural Networks*, 1995, **6**(1): 94～104
13 Zhang S W, Constantinides A G. Lagrange programming neural networks. *IEEE Transactions on Circuits and Systems*, 1992, **39**(7): 441～452
14 Jun W, Vira C. Recurrent neural networks for linear programming: Analysis and design principles. *Computers & Operations Research*, 1992, **19**(3/4): 297～311
15 Xia Y-S, Wu X-Y. An augmented neural networks for solving linear and quadratic programming problems. *Acta Electronica Sinica*, 1995, **23**(1):67～72 (in Chinese)
16 Xia Y S. A new neural networks for solving linear programming problems and its application. *IEEE Transactions on Neural Networks*, 1996, **7**(3): 525～527
17 Xia Y S. A new neural networks for solving linear programming problems and quadratic programming problems. *IEEE Transaction on Neural Networks*, 1996, **7**(12): 1544～1547
18 Leung Y, Chen K Z, Jiao Y C, Gao X B, Leung K S. A new gradient-based neural network for solving linear and quadratic programming problems. *IEEE Transaction on Neural Networks*, 2001, **12**(5): 1074～1083
19 TIAN Da-Gang, FEI Qi. New algorithm for linear programming with neural networks. *Acta Automatica Sinica*, 1999, **25**(5): 709～712(in Chinese)
20 Tian D G, Fei Q. An extension of the entropic perturbation method of linear programming. *MMOR(ZOR)*, 1999, **50**(1): 17～25
21 MA Zhong-Fan. The Newest Advances in Linear Programming. Beijing: Science Press, 1994. 7～8(in Chinese)
22 Fang S C, Tsao H S T. Linear programming with entropic perturbation. *ZOR*, 1993, **37**(2): 171～186

TIAN Da-Gang Received his bachelor degree in mathematics from Nanchang University, P. R. China, master degree in applied mathematics from Wuhan University, P. R. China and the Ph. D. degree in engineering from Huazhong University of Science and Technology, P. R. China, in 1999. He is currently an associate professor at the college of management, University of Shanghai Sciences and Technology, Shanghai. His research interests include decision theory, decision support system, and evolution computing.

对偶性和线性规划问题的神经网络解法

田大钢
（上海理工大学管理学院　上海　200093）
（E-mail: ly008369@online.sh.cn）

摘　要　通过一种新的对偶形式，得到一种新的易于实现的解线性规划问题的神经网络，证明了网络具有全局指数收敛性，使得线性规划问题的神经网络解法趋于完善.

关键词　对偶性，神经网络，线性规划
中图分类号　TP18;O22