Synthesis of a \textit{k}-winners-take-all neural network using linear programming with bounded variables

L. V. Ferreira, E. Kaszkurewicz and A. Bhaya

Department of Electrical Engineering, NACAD–COPPE/Federal University of Rio de Janeiro
P.O Box 68504, 21945-970, Rio de Janeiro, RJ, BRAZIL
E-mail: {lvalente,eugenius}@coep.ufrj.br, amit@nacad.ufrj.br

Abstract—A \textit{k}-winners-take-all (KWTA) problem is formulated as a linear programming (LP) problem with bounded variables. The solution set of the LP problem determines the winners. The LP problem is converted into an unconstrained optimization problem with two exact penalty functions, that is solved by using a gradient descent method implemented as a neural network. Theoretical results ensuring the convergence to the correct solution are provided.

I. INTRODUCTION

The \textit{k}-winners-take-all problem is that of determining the \textit{k} largest components of a given vector \( \mathbf{c} \in \mathbb{R}^n \). This problem appears in competitive network learning and pattern recognition. Many networks have been proposed to solve this problem, and these networks are referred as KWTA networks.

To solve this problem we propose a KWTA neural network designed on the basis of an LP problem with bounded variables, using the penalty function method. A nonsmooth computational energy function is associated to this LP problem. This converts the original LP problem into an unconstrained nonlinear optimization problem. This new optimization problem is solved by a gradient system, which represents the KWTA neural network that, in turn, minimizes the given energy function.

Convergence conditions for this system are obtained through an analysis that uses nonsmooth diagonal type Lyapunov functions and a gradient system represented in Persidskii type form [1], [2].

This formulation is inspired partially by the one presented by Urahama & Nagao [3], where this problem is formulated as an integer programming problem, which is mapped into a nonlinear programming problem, and solved by minimizing an associated Lagrangian function.

We modify the latter approach as follows. Instead of using the integer programming approach proposed in [3], we relax it to a LP problem with variables confined to the interval \([0,1]\). An exact penalty method applied to the relaxed LP problem produces a so-called computational energy function. The latter is minimized using a gradient (descent) system. Compared with the approach proposed in [3], the approach using linear programming has advantages. The gradient of the energy function provided by the penalty method is easily derived, leading to the development of simple convergence conditions. In addition, the corresponding neural circuit is very simple and can be implemented using resistors, amplifiers and switches. As suggested in [4], the transient time of this network can be adjusted by introducing a time scale factor in the model, making the proposed network suitable for real time operation. Moreover, simulation examples indicate that the network obtained presents better separability resolution than the circuit proposed in [3].

The penalty functions used are exact, thus ensuring that there exist finite values of the penalty parameters for which the exact solution of the LP problem can be found. The nondifferentiability of the penalty functions is not a drawback, since appropriate mathematical tools for dealing with this are available. We obtain constant penalty parameters that ensure global convergence of the network trajectories to the equilibrium state, which corresponds to the unique minimizer of the energy function.

The convergence analysis carried out here follows the same framework presented in [5] and [6] by the present authors, i.e., diagonal type Lyapunov functions are used and the systems of differential equations are described, equivalently, in Persidskii type form. Persidskii type systems and diagonal type functions are discussed in [1] and [2]. For reasons of space, some proofs are omitted and can be found in the technical report [7].

Utkin [8] also uses exact penalty functions and sliding modes in the design of gradient dynamical systems that solve optimization problems; his analysis is carried out by using the so-called equivalent control method. In this context, the contribution of this paper is a Lyapunov function analysis of the same class of gradient dynamical systems, leading to simple conditions on the penalty parameters for global convergence to the optimum.

In this paper we denote column vectors by boldface lowercase letters like \( \mathbf{c} \). Scalars are represented by lowercase italic letters, like \( k \) and \( n \). Matrices are denoted by uppercase boldface letters, like \( \mathbf{P} \). Vector functions are denoted by \( f(\mathbf{x}) \) which, unless otherwise specified,
are diagonal type functions, i.e.,

\[ f(x) = (f(x_1), \ldots, f(x_n))^T, \]

for \( j = 1, \ldots, n, x_j \) are the components of vector \( x \).

II. THE LP PROBLEM

Consider the following LP problem with bounded variables:

\[
\begin{align*}
\text{Maximize } & \quad z = c^Tx \\
\text{Subject to } & \quad 1x = k \\
& \quad 0 \leq x \leq 1^T.
\end{align*}
\]

where \( c = [c_1, \ldots, c_n]^T \), \( 1 = [1, \ldots, 1] \in \mathbb{R}^{1 \times n} \), \( k \leq n \in \mathbb{N} \) is a nonnegative integer and \( x \in \mathbb{R}^{n \times 1} \).

Notice that the LP problem (1) is always feasible. Moreover, the feasible set is closed and bounded, which ensures that its solution set is limited and bounded.

A solution of the LP problem is a vector \( x^* \) such that \( x^* = \text{argmax}(z) \), where \( x^* \in \{ x : 1x = k \} \cap \{ x : x_i \in [0,1], i = 1, \ldots, n \} \).

If the components of vector \( c \) are distinct, the solution of (1) possesses two properties that make the synthesis of a KWTA neural network possible:

1. The solution \( x^* \) of the LP problem (1) has \( k \) components equal to 1 and \( n - k \) components equal to 0.
2. The \( k \) nonzero components of the solution \( x^* \) correspond exactly to the \( k \) largest components of the vector \( c \) of the objective function.

This can be summarized in the following proposition.

**Proposition 1:** Consider the LP problem (1), and let the components of vector \( c \) be distinct. Then, the solution of the LP problem (1) is unique and presents \( k \) components equal to one, which multiply the \( k \) largest components of vector \( c \) in the objective function \( z \), and the \( n - k \) remaining components are equal to zero.

**Proof:** Let \( C \) be the set of the \( k \) largest components of vector \( c \) and define the sets of indices \( I := \{ i : c_i \in C \} \) and \( J := \{ i : c_i \notin C \} \). Without loss of generality, let \( c_1, c_2, \ldots, c_k \) be the \( k \) largest components in vector \( c \).

The feasible set of problem (1) corresponds to a hyperplane \( \Pi := \{ x : \sum_{p=1}^n x_p = k \} \) inscribed in the hypercube defined by \( H := \{ x : x_p \in [0,1], p = 1, \ldots, n \} \). The vertices of the hypercube \( H \) intersected by hyperplane \( \Pi \) are only those ones that present \( k \) components equal to one and the remaining \( n - k \) components equal to zero. Moreover, no edges of the hypercube are cut by the hyperplane \( \Pi \), this is so because given the fact that each point on an edge of the hypercube \( H \) is described by a component \( x_q \in [0,1] \), for some index \( q \), and the remaining \( n - 1 \) components \( x_p \in [0,1] \), for \( p \neq q \), then, for such points the sum \( \sum_{p=1}^n x_p \) does not yield an integer value. Thus, for \( k \in \mathbb{N} \) the hyperplane \( H \) does not cut any edges of the hypercube \( H \).

Since the coefficients \( c_p, p = 1, \ldots, n \) are distinct, the hyperplane described by the objective function of the LP problem (1) is not parallel to the hyperplane \( \Pi \). This implies that the solution of the LP problem (1) is unique and is one of the vertices of the hypercube \( H \) intersected by the hyperplane \( \Pi \), then it is a vector of \( k \) ones and \( n - k \) zeros. Consequently, since \( c_i > c_j \), for all \( i \in I \) and \( j \in J \), it is immediate that the vector that maximizes \( z \), while satisfying the constraints of the LP problem, is \( x^*_i = 1 \), for \( i \in I \) and \( x^*_j = 0 \), for \( j \in J \). Then, the vector \( x^* \) that solves the LP problem (1) is a vector, such that its \( k \) components equal to 1 correspond to the \( k \) largest components of vector \( c \) and the remaining components are equal to zero. This concludes the proof.

Comparing with the formulation using integer programming considered in [3], that is obtained by replacing the constraints on vector \( x \), in problem (1), with \( x_i \in \{0,1\}, i = 1, \ldots, n \). Proposition 1 states that LP problem (1) and the integer programming problem have the same solution \( x^* \).

Based on the preceding proposition, we can build a neural circuit that solves the LP problem (1), or, in other words, design a neural classifier that selects the \( k \) largest signals from vector \( c \). Such a network is presented in the next section and it is referred as a KWTA network.

III. MATHEMATICAL FORMULATION OF THE KWTA NEURAL NETWORK

Our aim, in this section, is to synthesize a neural network that solves problem (1). We use the penalty function method [9] with two exact penalty functions, each one corresponding to a different set of constraints of this LP problem.

After converting problem (1) into a minimization problem, by means of the penalty function method, we obtain the associated unconstrained optimization problem:

\[
\text{Minimize } E(x, \gamma, \rho) = -c^Tx - \gamma \sum_{j=1}^n \min(0, x_j) + \gamma \sum_{i=1}^n x_i^+ + \rho |1x - k| \tag{2}
\]

where, for each \( j \)

\[
x_i^+ = \begin{cases} 
  x_j & \text{if } x_j > 1 \\
  0 & \text{if } x_j \leq 1.
\end{cases}
\]

Observe that \( E \) is constituted by the objective function of the original problem plus two penalty terms, each one acting on a different set of constraints.

According to the penalty function method, there exist finite values of \( \gamma \) and \( \rho \), for which the solution of problem (2) corresponds exactly to the solution of problem (1).
The objective function $E$ of problem (2) is convex, which ensures that $E$ has a unique global minimum and that its minimizer can be found by means of a gradient descent method [4], [9]. Consider the gradient system $\dot{x} = -\nabla E(x)$, that minimizes $E$, which is given by:

$$\dot{x} = c - \gamma \left[ \text{hsgn}(x) + \text{uhsgn}(x) \right] - \rho 1^T \text{sgn}(1x - k) \quad (3)$$

where for each $j$:

$$\text{hsgn}(x_j) = \begin{cases} 
1 & \text{if } x_j < 0 \\
0 & \text{if } x_j > 0
\end{cases}$$

$$\text{uhsgn}(x_j) = \begin{cases} 
0 & \text{if } x_j < 1 \\
1 & \text{if } x_j > 1
\end{cases}$$

$$\text{sgn}(1x - k) = \begin{cases} 
-1 & \text{if } 1x - k < 0 \\
1 & \text{if } 1x - k > 0
\end{cases}$$

The system of differential equations (3) describes a modified Hopfield network that, for $\gamma$ and $\rho$ sufficiently large, is capable of selecting the $k$ largest signals from the vector $c$, which are referred to as the winners. The functional block diagram of the network is depicted in figure 1.

The network, shown in figure 1, consists of a parallel array of neurons, which receive, as input signals, the output signal of a perceptron neuron with synaptic weights equal to unity and the penalty parameters, $\gamma$ and $\rho$, can be interpreted as network gains. The $k$ largest input signals $c_p$, where $p$ is the index of each of the $k$ winners, are indicated by the $k$ neurons that remain active.

Fig. 1. Neural circuit described by the system of equations (3).

Two different terms can be identified in equation (3), each one acting on a corresponding set of constraints:

$$l_1 = \rho 1^T \text{sgn}(1x - k)$$

$$l_2 = \gamma \left[ \text{hsgn}(x) + \text{uhsgn}(x) \right]$$

For $\gamma$ and $\rho$ sufficiently large, the term $l_1$ is responsible for keeping the trajectories within the set $\Delta := \{ x : 1x - k = 0 \}$. At the same time, the term $l_2$ is responsible for maintaining trajectories within the intersection of the sets $\{ x : x_j \in [0, 1] \}$ and $\Delta$.

Since for $\gamma$ and $\rho$ sufficiently large the trajectories of (3) converge to the solution of the LP problem (1), which satisfies Proposition (1), the circuit modeled by (3) presents KWTA property.

Notice also that the right-hand side of system (3) is discontinuous. This is due to the presence of exact penalty functions in the energy function (2), which are nondifferentiable at the feasible set of the LP problem. For this reason, solutions of (3) are considered in the sense of Filippov [10].

Some comments are necessary about the formulation of LP problem (1) and its consequences in the design of the neural network modeled by (3). Urahama & Nagao [3] propose an integer programming problem of the same type as (1), with $x \in \{0, 1\}^n$, which means that each component of vector $x$ is 0 or 1.

Such a formulation is also possible with our model. We can regard the LP problem (1) as a problem with three equality constraints, and the penalty functions for this problem are $\gamma_1 \|x\|_1$, $\gamma_2 \|x - 1\|_1$ and $\rho \|1x - k\|$, where $\gamma_1 \neq \gamma_2$. The introduction of the parameter $\gamma_2$ is necessary, for if $\gamma_1 = \gamma_2$, the violation of the constraint $x \in \{0, 1\}^n$ is not penalized for variables in the interval $[0, 1]$. However, the introduction of two penalty functions with different penalty parameters, in order to penalize the violation of the constraint $x \in \{0, 1\}$, does not bring any benefits to the analysis and implementation of the neural network modeled by (3). The introduction of an extra parameter makes convergence analysis more complex, since it also has to be adjusted.

As far as practical implementation of the neural circuit is concerned, as suggested in [11], a large number of connections constitutes a major bottleneck for VLSI CMOS implementation of such neural networks, and an extra parameter, which is interpreted as a network gain, demands at least $n$ extra components plus wiring to connect these components. Thus the use of linear programming with bounded variables is more convenient than using integer programming.

IV. Convergence analysis

The presence of discontinuous functions in the system of equations (3) requires that convergence of its trajectories to be understood in the sense of sliding modes. Once the trajectories reach the LP problem feasible set, a sliding motion is initiated and its trajectories "chatter" within a neighbourhood of the unique minimizer of the function $E$, where they remain confined. The "chattering" phenomenon is described as a high frequency switching around the feasible set of the LP problem, which
is the surface of discontinuity of the dynamical system described by (3). Detailed expositions concerning differential equation with discontinuous righthand sides and theory of sliding modes can be found, for instance, in [10], [12], [8], [13], [14], [15].

Due to the phenomenon of chattering, the trajectories are said to converge to the equilibrium set of system (3), which consists of a neighbourhood of the solution $x^*$ of the optimization problem (2).

Convergence analysis of the system described by equation (3) is divided into two steps. First, we prove convergence to the feasible set of problem (1), which is given by the intersection

$$
\Omega := \Delta \cap \Phi \quad (4)
$$

where $\Delta := \{x : 1x - k = 0\}$ and $\Phi := \{x : x_j \in [0,1], \text{ for each } j\}$.

Once in the feasible set, it is necessary to prove that the trajectories converge to the solution of problem (1). This analysis results in sufficient conditions that the penalty parameters $\rho$ and $\gamma$ must satisfy in order to ensure convergence of the trajectories of equation (3) to the solution set of the LP problem (1). Equivalently, this analysis provides guidelines to set the gains $\rho$ and $\gamma$ such that the solution of LP problem (1) belongs to the equilibrium set of (3), and to guarantee that the trajectories converge globally to this set and remain in it thereafter.

Convergence results are obtained using a Lyapunov function $V$. It is possible to show finite time convergence; this means, for example, that there exists $\varepsilon > 0$, such that $\dot{V} < -\varepsilon$. For a detailed exposition on Lyapunov theory see, for instance, [16], [17], [14].

The following lemma states sufficient conditions that ensure convergence of the trajectories of system (3) to the feasible set of problem (1).

Lemma 1: Consider the system of ordinary differential equations (3) and assume that it admits solutions in the sense of Filippov. For any initial conditions, if $\gamma$ and $\rho$ satisfy

$$
\rho > \frac{\|c\|^2}{\sqrt{n}} + \gamma \quad (5)
$$

$$
\gamma > \frac{n}{2} \|c\|^2 \quad (6)
$$

then the trajectories of system (3) reach the set $\Omega$, defined in (4), in finite time and remain in this set thereafter.

Proof: For brevity, we present only the basic ideas of this proof. See complete proof in [7]. Let $\nu$ be the subgradient of $P_\phi(x) = \sum_{j=1}^n \min(0, x_j) + x_j^+$. Premultiplying system (3) by vector $1$ we get:

$$
\dot{r} = 1c - \gamma 1\nu(x) - \rho 11^T \text{ sgn}(r). \quad (7)
$$

Notice that the equations (3) and (7) form the following Persidskii type system [1], [2], [5], written in vector notation:

$$
\dot{x} = \left( \begin{array}{c} c \\ 1 \end{array} \right) - \left( \begin{array}{c} I_n \\ 11^T \end{array} \right) \left( \begin{array}{c} \gamma \nu(x) \\ \rho \text{ sgn}(r) \end{array} \right) \quad (8)
$$

where $I_n$ denotes the identity matrix of order $n$.

Consider also the following nonsmooth diagonal type lemma candidate Lyapunov function associated to this system:

$$
V(x,r) = \gamma \sum_{j=1}^n \int_0^{x_j} \nu(t)dt + \rho \int_0^r \text{ sgn}(t)dt. \quad (9)
$$

Due to the presence of discontinuous terms in the righthand side of (3) the proof is divided into two steps. In step 1 we suppose that the trajectories do not belong to the set $\Delta = \{x : 1x = k\}$, which means that no sliding motion occurs and the solutions of system (3) are considered in the usual sense. In step 2 we consider $x \in \Delta$, in this case sliding motion occurs, and the solutions are considered in the sense of Filippov. Step 1: $x \notin \Delta$. Then, the time derivative of function (9) along trajectories of system (8) is given by:

$$
\dot{V} = -\|g(x)\|^2 + \|g(x)\|^T c, \quad (10)
$$

where $g(x) = \gamma(\text{hsgn}(x) + \text{ulsgn}(x))$.

Once the gain $\rho$ satisfy inequality (5), we assure $\dot{V} < 0$ and this concludes the first step.

Step 2: If $x \in \Delta$ and $x_j \notin \Phi$, for some $j$. In this case, function (9) reduces to:

$$
V(x) = \sum_{j=1}^n \int_0^{x_j} h(t)dt. \quad (11)
$$

The time derivative of function (11) is given by:

$$
\dot{V} = \nabla V^T \dot{x} = \nu(x)^T P(c - \gamma \nu(x)), \quad (12)
$$

where $P$ is the projection matrix onto the null space of the row vector $1$.

Inequality (6) being satisfied with the corresponding parameter (gain) $\gamma$ we assure $\dot{V} < 0$. This concludes step 2. It is easy to show that the trajectories of system (3) converge to the feasible set of the LP problem (1) in finite time and remain in it thereafter.

Notice that when trajectories are confined to the set $\Omega$, the penalty terms of the energy function (2) go to zero and it is reduced to the objective function of the original LP problem, described by (1).

Lemma 1 says that for parameters $\rho$ and $\gamma$ satisfying inequalities (5) and (6), respectively, the system trajectories reach the feasible set of LP problem (1), but it is still necessary to prove two other auxiliary results – first we must show that if $\rho$ and $\gamma$ satisfy the bounds of lemma 1, then the LP problem (1) and the unconstrained problem (2) have the same solution $x^*$; this is necessary in
order to validate our arguments, we are proposing a neural classifier on the basis of the unique solution of LP problem (1), so we need the network modeled by (3) to converge to the solution set of the original LP problem. Second, it is necessary to show that, if $\gamma$ and $\rho$ satisfy lemma 1, then the system trajectories converge to the solution of the LP problem (1); this is necessary because lemma 1 does not ensure convergence to the solution of the LP problem (1), but to its feasible set. This is taken care of in the following lemmas.

**Lemma 2:** If the network gains $\rho$ and $\gamma$ satisfy the bounds of lemma 1, then the LP problem (1) and the unconstrained problem (2) have the same solution.

**Proof:** Since $E$ is convex for every $x$, a local minimum of $E$ is, in fact, a global minimum. We only need to prove that the minimizer of $E$ belongs to $\Omega$, since $E(x) = z$, $\forall x \in \Omega$, where $\Omega$ is defined in (4).

We prove this lemma by contradiction. Let $x^* \notin \Omega$ be the minimizer of $E$ and suppose that $\rho$ and $\gamma$ satisfy the bounds of lemma 1. Then, there exists at least one trajectory of system (3) that does not belong to $\Omega$, which is a contradiction, since by lemma 1 all the trajectories of system (3) converge to $\Omega$ in finite time and remain in it thereafter. Thus $x^* \in \Omega$, concluding the proof. □

This lemma ensures that the LP problem (1) and the problem of minimizing $E$ have the same solution set, provided that $\rho$ and $\gamma$ satisfy the bounds of lemma 1. It remains to be shown that the trajectories of (3) converge to the solution of the LP problem, provided that the conditions of lemma 1 are verified. This result is stated in the next lemma.

**Lemma 3:** If the network gains $\rho$ and $\gamma$ satisfy the bounds established in lemma 1, then the trajectories of system (3) reach the solution of the LP problem (1) in finite time and remain in it thereafter.

**Proof:** See reference [7]. □

The previous three lemmas complete our convergence analysis. The key result is lemma 1 and all the other lemmas rely on it. Lemma 2 ensures that the minimizer of the computational energy function $E$, given in (2) is the maximizer of the LP problem (1), and lemma 3 states that the trajectories of system (3) converge to the solution of the LP problem. These results, together with proposition 1, ensure that the neural network modeled by the gradient system (3) is a $k$-winners-take-all network. We now have all the elements needed to enumerate the theorem below, which is the main result of this paper.

**Theorem 1:** Consider the neural network modeled by the system of ordinary differential equations (3) and assume that the network gains $\gamma$ and $\rho$ satisfy the bounds of lemma 1. Then, given a vector $c \in \mathbb{R}^{n \times 1}$, with distinct components, and a positive integer $k \leq n$, the neural network described by (3) is a KWTA network.

This Theorem states that our KWTA network is capable of extracting $k$ winners from a given vector $c$. Based on the fact that the solution of the LP problem (1) satisfies Proposition 1, convergence of the trajectories of (3) to the solution of the LP problem (1) provides our circuit with a KWTA property.

The bounds stated by Lemma 1 are simple and their derivation is straightforward. This occurs for two main reasons: (i) the use of an adequate diagonal type Lyapunov functions and Persidskii type system representation, and (ii) the use of exact penalty functions with analytical expressions for their gradients.

In order to illustrate the theoretical results presented in this paper, in the next section we present some examples with the corresponding computer simulations.

## V. Simulation examples

In this section we present some examples to illustrate the theoretical results presented in the previous sections.

**Example 1:** Consider the vector

$$c = (1.23 \quad 1.32 \quad 1.25 \quad 1.47)$$

from which we wish to extract the two largest signals, that is $k = 2$.

Using Theorem 1 the bounds for the penalty parameters are $\gamma > 5.28$ and $\rho > 6.61$. Let $\gamma = 5.30$ and $\rho = 6.70$.

The simulation results are shown in figure 2, with initial conditions chosen (arbitrarily) at the origin. The components $x_2 = x_4 = 1$ correspond to the two largest components of vector $c$ – the winners; and the remaining ones, $x_3 = x_4 = 0$, correspond to the losers.

In this example, in order to have faster convergence, we used a time scale factor $\mu = 10$ which was chosen empirically and is introduced in equation (3), which takes the form $\dot{x} = -\mu \nabla E(x)$. Other techniques for choosing $\mu$ are discussed in [4, Section 3.1.4].

![Fig. 2. Trajectories of example (1), showing correct classification of the two winners.](image-url)
**Theorem 1**, we get

\[ c \text{ for } j = 1, \ldots, 9 \text{ and circuit outputs are evaluated for } c_{10} \text{ varying from 0 to 5 V. For this example, let us consider } c_{10} = 3.3 V, \text{ that is:}

\[
e = (0.5, 1, 1.5, 2.5, 3, 3.5, 4, 4.5, 3.3)
\]

In this case, the winners are \( c_7, c_8 \) and \( c_9 \). Using **Theorem 1**, we get \( \gamma > 45.32 \) and \( \rho > 48.19 \). Let \( \gamma = 45.35 \) and \( \rho = 48.20 \). As in the previous example, in order to accelerate convergence, we used a time scale factor \( \mu = 10 \). Simulation results are shown in figure 3.

**Example 2**: (Adapted from [3, page 777]) In [3] it is considered a circuit with \( n = 10 \) and \( k = 3 \), where \( c_j = j/2 \) V, for \( j = 1, \ldots, 9 \) and circuit outputs are evaluated for \( c_{10} \) varying from 0 to 5 V. For this example, let us consider \( c_{10} = 3.3 V \), that is:

\[
e = (0.5, 1, 1.5, 2.5, 3, 3.5, 4, 4.5, 3.3)
\]

In this case, the winners are \( c_7, c_8 \) and \( c_9 \). Using **Theorem 1**, we get \( \gamma > 45.32 \) and \( \rho > 48.19 \). Let \( \gamma = 45.35 \) and \( \rho = 48.20 \). As in the previous example, in order to accelerate convergence, we used a time scale factor \( \mu = 10 \). Simulation results are shown in figure 3.

![Fig. 3. Trajectories of example (2), showing KWTA behaviour.](image)

Urahama & Nagao [3] show that, for this example, their circuit has resolution limit is 0.5 V and if the difference between the smallest winner and the largest loser is less than this limit, like in this example, then the outputs of their circuit become fuzzy. As shown for this case, our network produces correct classification of the entries of vector \( e \), showing a better separability resolution.

In the preceding examples, the discontinuous terms in equation (3) cause the trajectories shown in graphics 2 and 3, to chatter in a neighbourhood of the feasible set of the LP problem. Chattering happens when trajectories converge to the surface of discontinuity of system (3), which consists of the whole feasible set of the LP problem (1).

**VI. CONCLUDING REMARKS**

In this paper we propose a new \( k \)-winners-take-all neural network designed on the basis of a linear programming framework with bounded variables. The theoretical formulation of the network is based on the fact that the solution set of the LP problem can be designed to coincide with the equilibrium set of a KWTA gradient type neural network.

Convergence analysis proceeds by representing the system of differential equations that models the network in a suitable Persidskii-like form, and using a diagonal type Lyapunov function.

Compared with the network based on a nonlinear programming relaxation framework proposed in [3], our network has the advantage of being synthesized from a simple linear programming problem. Performance of our network depends on the correct adjustment of the parameters \( \gamma \) and \( \rho \), which, however have to be calculated only once.

Urahama & Nagao [3] claim that their model achieves almost instantaneous convergence with no transient phase, as opposed to Hopfield type models. However, we have shown that our network provides convergence in finite time. Also, Cichocki & Unbehauen [4] claim that in Hopfield type models the transient can be made shorter by choosing appropriate time scales, which can be chosen in such a way that the convergence time can be determined a priori. This is also the case for our network, making it suitable for real time operation.

**REFERENCES**


