An Augmented Lagrangian Neural Network for the Fixed-Time Solution of Linear Programming

Dayanna T. Toro and José M. Lozano
Department of Electrical Engineering
Universidad de Guanajuato
Carretera Salamanca - Valle de Santiago, km 3.5 + 1.8
Comunidad de Palo Blanco. P.C. 36885, Salamanca, México.
Email: dtesalia@ugto.mx, jm.lozano@ugto.mx

Juan Diego Sánchez-Torres
Research Laboratory on Optimal Design, Devices and Advanced Materials -OPTIMA-
ITESO University
Tlaquepaque, México
dsanchez@iteso.mx

***Abstract*—In this paper, a recurrent neural network is proposed using the augmented Lagrangian method for solving linear programming problems. The design of this neural network is based on the Karush-Kuhn-Tucker (KKT) optimality conditions and on a function that guarantees fixed-time convergence. With this aim, the use of slack variables allows transforming the initial linear programming problem into an equivalent one which only contains equality constraints. Posteriorly, the activation functions of the neural network are designed as fixed time controllers to meet KKT optimality conditions. Simulations results in an academic example and an application example show the effectiveness of the neural network.**

I. Introduction

Linear programming is an important field of optimization, since many applications can be formulated and solved through a linear representation, which has generated major research within this area. Over the years, for many applications, a variety of numerical algorithms have been developed, being Pyne one of the first to introduce the use of dynamic systems to solve optimization problems. The main advantage of this technique is that the system constantly seeks new solutions as the parameters of the problems are varied [1].

Since then, many extensions of these systems have been developed in which recurrent neural networks have received substantial attention. In 1986, Tank and Hopfield proposed the first recurrent neural network for solving linear programming problems [2], which inspired the development of numerous neural networks in the area of optimization. In 1988, Kennedy and Chua [3] proposed a neural network with a finite penalty parameter for nonlinear programming, with the disadvantage that the penalty parameter had to grow infinitely to achieve convergence. To avoid using the penalty parameter, others methods such as Lagrange multiplier were introduced [4].

In this sense, Wang [5] proposed a recurrent neural network with a time-varying threshold vector to solve linear programming problems, with the disadvantage that the proposed network is asymptotically stable. Similarly, it was shown [6] that the use of differential algebra, in conjunction with KKT conditions, has a faster convergence than other methods used in linear programming; other investigators [7]–[9] also use recurrent neural networks to study the convergence of a linear problem, obtaining the optimal solution of a system in finite time.

In the same way, investigations have also been carried out in nonlinear systems with discontinuous activation function [10], [11]. Similarly, in later studies a neural network was proposed that solved problems where the objective function may or may not be continuously differentiable [12] and a neural network capable of solving non-convex problems [13] was successfully proposed, which expanded the universe of systems that can be solved.

Other studies have used the augmented Lagrangian method for solving optimization problems, achieving better numerical stability [14]. For this reason, the aim of this paper is to develop a recurrent neural network based on Lagrangian augmented and slack variables to solve linear programming problems, thus illustrating its operation in a classic optimization problem.

The remainder of this paper is organized as follows: in Section II, the preliminaries related to the development of the neural network are presented. Section III describes the model of the recurrent neural network and its performance through an academic example. In Section IV an application is presented were the energy that is delivered by the different generation units of a microgrid are maximized. Finally, conclusions are given in Section V.

II. Mathematical Preliminaries

In this section, the important concepts for the development of the network structure are presented.

A. Slack Variables

Proposition 1. *If a and b are two real numbers, then $a \leq b$ if and only if $a + y^2 = b$ for some number y [15].*

A classical optimization problem usually has both equality and inequality constraints that, when combined, often interact in complex ways, so that slack variables can be used to transform the original problem into a problem that only has equality restrictions.

Now, consider the general programming problem

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{h} \end{aligned} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the vector of decision variables, $\mathbf{c}$, $\mathbf{l}$, $\mathbf{h} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a full row-rank matrix ($\text{rank}(A) = m, m \leq n$).

The general problem (1) can be transformed into a problem of equality constraints by well-knows techniques such as adding *slack variables*. For this to occur, first the inequality is converted into independent inequalities, as follows

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.a.} \quad & \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \\ & -\mathbf{x} \leq -\mathbf{l} \\ & \mathbf{x} \leq \mathbf{h} \end{aligned} \tag{2}$$

Note that problem (2) is equivalent to problem (1); once this form is obtained, the *slack variables* are introduced in order to eliminate the inequality

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \\ & -\mathbf{x} + \mathbf{s}_1 + \mathbf{l} = \mathbf{0} \\ & \mathbf{x} + \mathbf{s}_2 - \mathbf{h} = \mathbf{0} \end{aligned}$$

where $\mathbf{s}_1 = \left[y_1^2, \ldots, y_n^2\right]$ and $\mathbf{s}_2 = \left[y_{n+1}^2, \ldots, y_{2n}^2\right]$ are the vectors that contain the slack variables.

When a problem has equality constraints, it can be written in vectorial form as shown below

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.a.} \quad & \mathbf{M}\mathbf{z} - \mathbf{d} = \mathbf{0} \end{aligned} \tag{3}$$

where

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{O}_{m\times n} & \mathbf{O}_{m\times n} \\ \mathbf{I}_{n\times n} & \mathbf{I}_{n\times n} & \mathbf{O}_{n\times n} \\ -\mathbf{I}_{n\times n} & \mathbf{O}_{n\times n} & \mathbf{I}_{n\times n} \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} \mathbf{b} \\ \mathbf{h} \\ -\mathbf{l} \end{bmatrix} \tag{4}$$

$$\mathbf{z} = \left[x_1, \ldots, x_n, y_1^2, \ldots, y_{2n}^2\right]^T \tag{5}$$

$\mathbf{A}$ is the matrix containing the equality constraints, $\mathbf{O}_{m\times n}$ is a matrix of zeros of dimension $m \times n$, $\mathbf{O}_{n\times n}$ is another matrix of zeros of dimension $n \times n$, $\mathbf{I}_{n\times n}$ is an identity matrix of dimension $n \times n$, $\mathbf{z}$ is the vector containing both the decision variables ($\mathbf{x}$) and the slack variables (y_i^2), which represent the additional value that needs to be added to satisfy the equality; each slack variables is elevated to the square to ensure that it is positive.

B. Augmented Lagrangian

The augmented Lagrangian method is a technique used to solve constraint optimization problems by transforming those problems into unconstrained equivalent ones. This method can be considered a hybrid between the method of the Lagrange multipliers and the penalty method, since it consists of adding an additional term to the Lagrangian.

Given $\boldsymbol{\sigma} = \mathbf{M}\mathbf{z} - \mathbf{d}$ and parting from problem (3), the augmented Lagrangian is defined as

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}) = \mathbf{c}^T\mathbf{x} + \boldsymbol{\lambda}^T\boldsymbol{\sigma} + \int_0^{\boldsymbol{\sigma}} \boldsymbol{\psi}(\boldsymbol{\sigma})d\mathbf{z} \tag{6}$$

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_k]^T$ represents the vector of the Lagrange multipliers and $\int_0^{\boldsymbol{\sigma}} \boldsymbol{\psi}(\boldsymbol{\sigma})d\mathbf{z}$ is the penalty function.

Usually, the penalty term most commonly used is $\frac{\rho}{2} \parallel \boldsymbol{\sigma} \parallel^2$, but in this work, a different function is used to illustrate its behavior.

III. Structure of the Network

A. Recurrent Neural Network Design

Starting from (6) and according to the KKT conditions [16], $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \mathbf{z}^*)$ is an optimal solution if

$$\begin{aligned} \nabla_x \mathcal{L}_\rho(\mathbf{x}^*, \mathbf{z}^*, \boldsymbol{\lambda}^*) &= \mathbf{0} \\ \nabla_y \mathcal{L}_\rho(\mathbf{x}^*, \mathbf{z}^*, \boldsymbol{\lambda}^*) &= \mathbf{0} \\ \nabla_\lambda \mathcal{L}_\rho(\mathbf{x}^*, \mathbf{z}^*, \boldsymbol{\lambda}^*) &= \mathbf{0} \end{aligned}$$

Subsequently, the following conditions must be fulfilled

$$\mathbf{c} + \frac{\partial \mathbf{z}}{\partial \mathbf{x}}\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\lambda} + \frac{\partial \mathbf{z}}{\partial \mathbf{x}}\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\psi}(\boldsymbol{\sigma}) = \mathbf{0} \tag{7}$$

$$\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\lambda} + \frac{\partial \mathbf{z}}{\partial \mathbf{y}}\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\psi}(\boldsymbol{\sigma}) = \mathbf{0} \tag{8}$$

$$\boldsymbol{\sigma} = \mathbf{0} \tag{9}$$

where $\mathbf{y} = [0 \ldots 0 \ y_1 \ y_2 \ \ldots \ y_{2n}]^T \in \mathbb{R}^k$ and $\mathbf{x} = [x_1 \ x_2 \ \ldots \ x_n \ 0 \ \ldots \ 0]^T \in \mathbb{R}^k$; this is done so they are of the same dimensions as $\mathbf{z}$, such that $\frac{\partial \mathbf{z}}{\partial \mathbf{y}}$ y $\frac{\partial \mathbf{z}}{\partial \mathbf{x}}$ are square matrices $k \times k$, of the form

$$\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_2}{\partial y_1} & \cdots & \frac{\partial z_k}{\partial y_1} \\ \frac{\partial z_1}{\partial y_2} & \frac{\partial z_2}{\partial y_2} & \cdots & \frac{\partial z_k}{\partial y_2} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial z_1}{\partial y_k} & \frac{\partial z_2}{\partial y_k} & \cdots & \frac{\partial z_k}{\partial y_k} \end{bmatrix}$$

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_2}{\partial x_1} & \cdots & \frac{\partial z_k}{\partial x_1} \\ \frac{\partial z_1}{\partial x_2} & \frac{\partial z_2}{\partial x_2} & \cdots & \frac{\partial z_k}{\partial x_2} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial z_1}{\partial x_k} & \frac{\partial z_2}{\partial x_k} & \cdots & \frac{\partial z_k}{\partial x_k} \end{bmatrix}.$$

The KKT conditions (7)-(9) are used to propose the following recurrent neural network in order to solve the linear program given in (3)

$$\dot{\mathbf{x}} = -\gamma\left[\mathbf{c} + \frac{\partial \mathbf{z}}{\partial \mathbf{x}}\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\psi}(\boldsymbol{\sigma}) + \frac{\partial \mathbf{z}}{\partial \mathbf{x}}\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\lambda}\right] \tag{10}$$

$$\dot{\mathbf{y}} = -\gamma\left[\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\psi}(\boldsymbol{\sigma}) + \frac{\partial \mathbf{z}}{\partial \mathbf{x}}\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\lambda}\right] \tag{11}$$

$$\dot{\boldsymbol{\lambda}} = \gamma\alpha\boldsymbol{\sigma} \tag{12}$$

where γ is a positive scaling constant, α is a nonnegative gain and $\psi(\boldsymbol{\sigma})$ is the derivative of the penalty function.

Given that $\boldsymbol{\sigma} = \mathbf{M}\mathbf{z} - \mathbf{d}$, then $\dot{\boldsymbol{\sigma}} = \mathbf{M}\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\dot{\mathbf{x}} + \mathbf{M}\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\dot{\mathbf{y}}$. Hence, from (10) and (11), it results

$$\begin{aligned}\dot{\boldsymbol{\sigma}} = &-\gamma\left(\mathbf{M}\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\mathbf{c} - \mathbf{M}\left[\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^2 + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^2\right]\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\lambda}\right) \\ &-\gamma\left(\mathbf{M}\left[\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^2 + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^2\right]\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\psi}(\boldsymbol{\sigma})\right).\end{aligned} \tag{13}$$

Selecting $\boldsymbol{\psi}(\boldsymbol{\sigma})$ in (13) as

$$\begin{aligned}\boldsymbol{\psi}(\boldsymbol{\sigma}) = &-\left(\gamma\mathbf{M}\left[\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^2 + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^2\right]\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\right)^{-1} \\ &\left[\gamma\left(\mathbf{M}\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\mathbf{c} + \mathbf{M}\left[\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^2 + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^2\right]\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\boldsymbol{\lambda}\right) - \boldsymbol{\phi}(\boldsymbol{\sigma})\right]\end{aligned} \tag{14}$$

the dynamics of $\boldsymbol{\sigma}$ in (13) reduces to $\dot{\boldsymbol{\sigma}} = -\boldsymbol{\phi}(\boldsymbol{\sigma})$.

In addition, when (14) is substituted in (7)-(9) is obtained:

$$\begin{aligned}\mathbf{c} + &\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\left\{\mathbf{M}\left[\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^2 + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^2\right]\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\right\}^{-1}\boldsymbol{\phi}(\boldsymbol{\sigma}) - \\ &\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\left[\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^2 + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^2\right]^{-1}\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\mathbf{c} = \mathbf{0}\end{aligned} \tag{15}$$

$$\begin{aligned}&\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\left\{\mathbf{M}\left[\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^2 + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^2\right]\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{z}}\right\}^{-1}\boldsymbol{\phi}(\boldsymbol{\sigma}) - \\ &\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\left[\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^2 + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^2\right]^{-1}\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\mathbf{c} = \mathbf{0}\end{aligned} \tag{16}$$

$$\boldsymbol{\sigma} = \mathbf{0}. \tag{17}$$

Since

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\left[\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^2 + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^2\right]^{-1}\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \mathbf{I} \tag{18}$$

$$\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\left[\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^2 + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^2\right]^{-1}\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \mathbf{0} \tag{19}$$

and $\boldsymbol{\phi}(\mathbf{0}) = \mathbf{0}$ it follows that the KKT conditions (15)-(16) are fulfilled when $\boldsymbol{\sigma} = \mathbf{0}$.

If $\boldsymbol{\phi}(\boldsymbol{\sigma})$ is defined as

$$\boldsymbol{\phi}(\boldsymbol{\sigma}) = k_1\frac{\boldsymbol{\sigma}}{\parallel \boldsymbol{\sigma} \parallel^{1/2}} + k_2\boldsymbol{\sigma} + k_3\boldsymbol{\sigma} \parallel \boldsymbol{\sigma} \parallel^{1/2} \tag{20}$$

with k_1, k_2 and k_3 positive gains, then the dynamics of $\boldsymbol{\sigma}$ in (13) is given by

$$\dot{\boldsymbol{\sigma}} = -k_1\frac{\boldsymbol{\sigma}}{\parallel \boldsymbol{\sigma} \parallel^{1/2}} - k_2\boldsymbol{\sigma} - k_3\boldsymbol{\sigma} \parallel \boldsymbol{\sigma} \parallel^{1/2}. \tag{21}$$

Finally, from (21), $\boldsymbol{\sigma} = \mathbf{0}$ in fixed time [17]. Therefore, from the KKT conditions (15)-(17), the linear programming problem (3) is solved in fixed time.

B. An Academic Example

Consider the linear programming problem [18]

$$\begin{aligned}\min \quad & 4x_1 + x_2 + 2x_3 \\ \text{s.t.} \quad & x_1 - 2x_2 + x_3 = 2 \\ & -x_1 + 2x_2 + x_3 = 1 \\ & -5 \leq x_1, x_2, x_3 \leq 5\end{aligned}$$

The proposed neural network (10)-(12) with the design function (14) and (20) is tested, with $k_1 = 8$, $k_2 = 3$, $k_3 = 3$, $\alpha = 1$, $\rho = 1$, $\gamma = 10^5$ and initial conditions are randomly selected within a range from -10 to 10. Results are shown in Fig.1 and Fig.2.

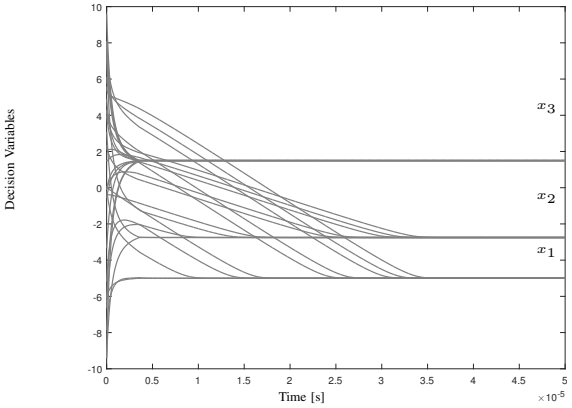

Fig. 1. Transient States of the Neural Network

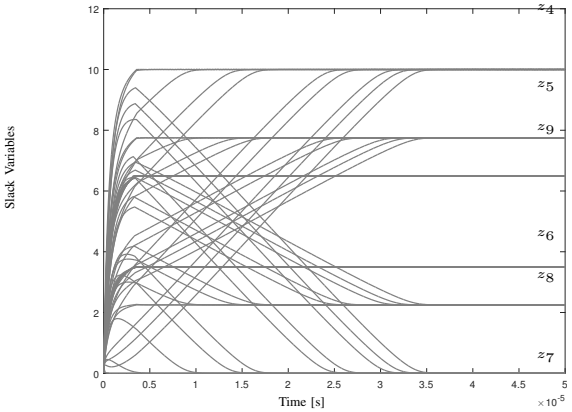

Fig. 2. Transient States of the Slack Variables

Hence, it can be observed that the network converges to the optimal solution $\mathbf{x}^* = [-5, -2.75, 1.5]$ with $\mathbf{z}^* = [-5, -2.75, 1.510, 7.75, 3.5, 0, 2.25, 6.5]$. Note that the first positions of the vector correspond to the decision variables $\mathbf{x}$. Fig. 3 shows the dynamics of $\boldsymbol{\sigma}$ which becomes zero in fixed time and satisfies the constraint.

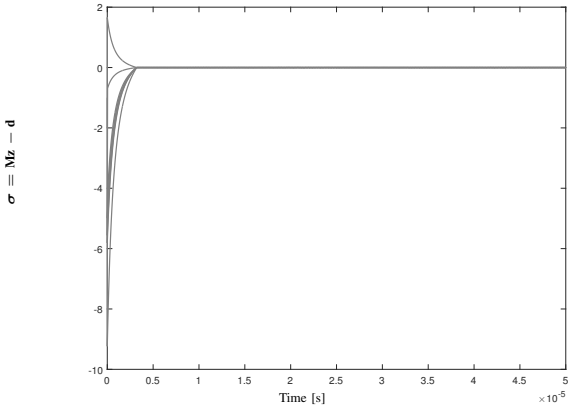

Fig. 3. Dynamics of $\boldsymbol{\sigma}$

IV. Application Example

An optimization problem is presented where the energy delivered by a non-autonomous microgrid (connected to the main system) must be minimized. The energy availability of each of the generation and storage units is considered, given that the demand required by the load is satisfied.

A general case of study considers both a wind and a solar power system as generation units, and also acknowledges a battery bank system as a storage unit, all connected to the main electrical system. This optimization problem can be expressed as follows:

$$\begin{aligned} \min \quad & P_G - P_W - P_S - P_B \\ \text{s.t.} \quad & P_G + P_W + P_S + P_B = P_L \\ & P_{Gmin} \leq P_G \leq P_{Gmax} \\ & P_{Wmin} \leq P_W \leq P_{Wmax} \\ & P_{Smin} \leq P_S \leq P_{Smax} \\ & P_{Bmin} \leq P_B \leq P_{Bmax} \end{aligned}$$

where $\mathbf{x} = [P_G \ P_W \ P_S \ P_B]$, $\mathbf{c} = [1 \ -1 \ -1 \ -1]$, $\mathbf{A} = [1 \ 1 \ 1 \ 1]$, $b = P_L$, $\mathbf{l} = [P_{Gmin} \ P_{Wmin} \ P_{Smin} \ P_{Bmin}]$ and $\mathbf{h} = [P_{Gmax} \ P_{Wmax} \ P_{Smax} \ P_{Bmax}]$.

In order to use the neural network (10)-(12), it is necessary to transform the problem into form (3):

$$\begin{aligned} \min \quad & P_G - P_W - P_S - P_B \\ \text{s.t.} \quad & P_G + P_W + P_S + P_B = P_L \\ & -P_G + y_1^2 = P_{Gmin} \\ & P_G + y_2^2 = P_{Gmax} \\ & -P_W + y_3^2 = P_{Wmin} \\ & P_W + y_4^2 = P_{Wmax} \\ & -P_S + y_5^2 = P_{Smin} \\ & P_S + y_6^2 = P_{Smax} \\ & -P_B + y_7^2 = P_{Bmin} \\ & P_B + y_8^2 = P_{Bmax} \end{aligned}$$

Once expressed in this way, the matrix $\mathbf{M}$ and the vectors $\mathbf{z}$ and $\mathbf{d}$ can be formed, as shown in (4) and (5). The maximum and minimum values defined in [18] are taken according to the available energy in an electrical microgrid prototype that was available at the time.

For this reason, $P_{Wmin} = 0$ and $P_{Wmax} = 240$ Watts were selected for the aero-generator. In the case of the photovoltaic panel, its minimum energy is equal to 0 Watts, while the maximum that the module is able to deliver is 1.2 Watts. On the other hand, the values of the battery were chosen to increase its lifespan as much as possible, reason for which P_{min} is set to 10% of its load and P_{Bmax} to 60%. Therefore, the main system is like an infinite bus, but in this work its maximum and minimum energy are respectively considered as 250 and 0 Watts.

To solve this problem, the proposed neural network (10)-(12) is used with $k_1 = 8$, $k_2 = 3$, $k_3 = 3$, $\alpha = 1$, $\rho = 1$, $\gamma = 10^4$, $P_L = 350$ Watts and the initial conditions are randomly selected within a range from 0 to 255 Watts. In Fig. 4 it can be seen how the decision variables converge to the optimal solution $\mathbf{x}^* = [36.8, 240, 1.2, 72]$ in fixed time. Fig. 5 shows the transient states of the slack variables until they reach stabilization in $\mathbf{z}^* = [36.8, 240, 1.2, 72, 213.2, 0, 0, 0, 36.8, 240, 1.2, 60]$.

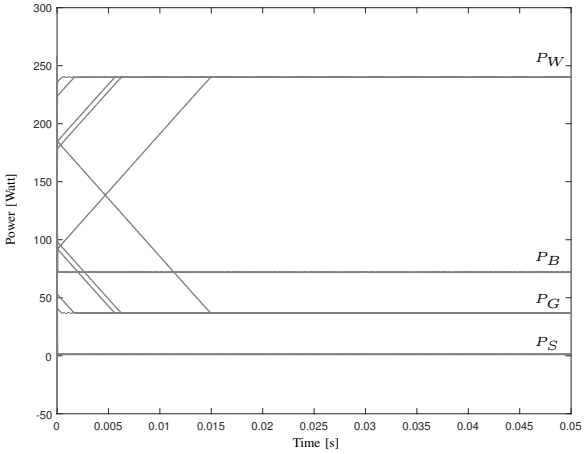

Fig. 4. Transient States of the Decision Variables

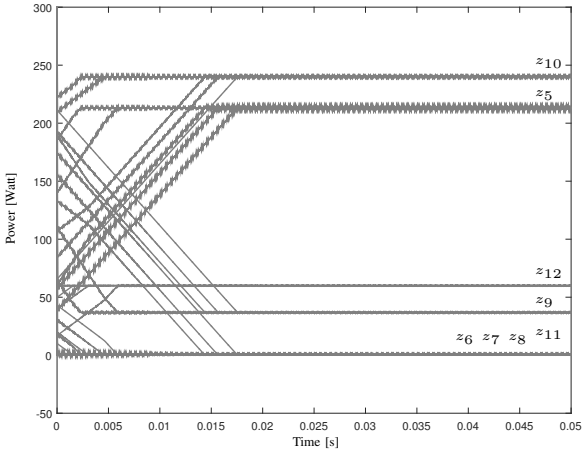

Fig. 5. Transient States of the Slack Variables

Subsequently, a variable load profile is used, as shown in Fig. 6, so the behavior of the proposed neural network can be observed when there are changes in the demand of the load. If the same characteristics mentioned above are used, the result obtained is presented in Fig. 7.

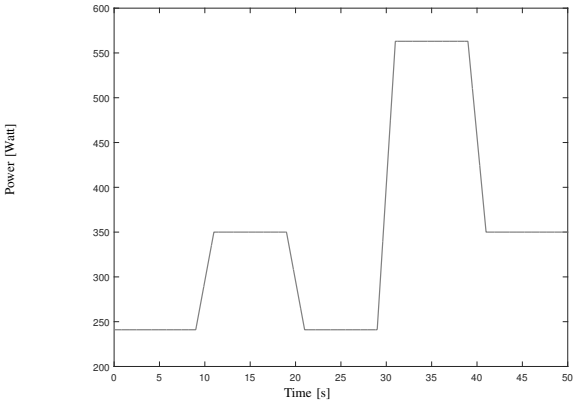

Fig. 6. Load Profile

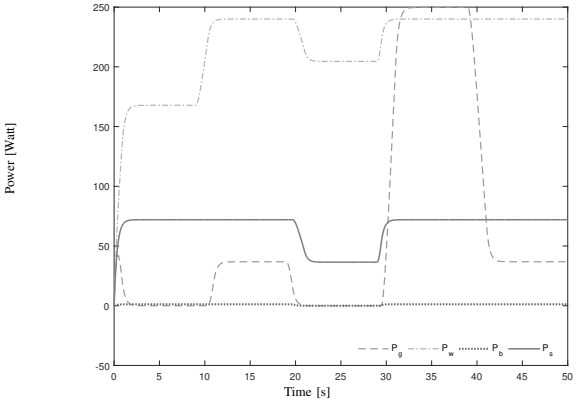

Fig. 7. DER's Power

When the load demand is low, the energy delivered by the main system is practically zero, since the energy supplied by the microgrid is sufficient to satisfy the demand. However, when the demand increases, the microgrid cannot provide enough energy, reason for which it takes what is additionally needed from the main system. At 30s, the demand of the load becomes so large that the system has to provide its maximum energy so that together with the microgrid, its load can be satisfied.

V. Conclusion

In this paper, a recurrent neural network for solving linear programming problems was proposed based on the augmented Lagrangian method and slack variables. The penalty function used was selected because of its capacity to provide fixed time convergence. Simulations results were presented to illustrate its performance on an academic example and on an application to distribute energy in an electrical microgrid. In the microgrid example, its power distribution maximized the use of the alternative power sources by minimizing the use of the main grid when a variable load profile was used. This shows that the proposed recurrent neural network operates successfully in real applications that can be defined as linear programming problems.

Acknowledgment

Dayanna Toro acknowledges CONACyT, México for the MSc scholarship number 718729.

References

[1] I. B. Pyne, "Linear programming on an electronic analogue computer," *American Institute of Electrical Engineers, Part I: Communication and Electronics, Transactions of the*, vol. 75, no. 2, pp. 139–143, 1956.

[2] D. Tank and J. Hopfield, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Transactions on Circuits and Systems*, vol. 33, no. 5, pp. 533–541, 1986.

[3] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 5, pp. 554–562, 1988.

[4] S. Zhang and A. G. Constantinides, "Lagrange programming neural networks," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, no. 7, pp. 441–452, Jul 1992.

[5] J. Wang, "Analysis and design of a recurrent neural network for linear programming," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 9, pp. 613–618, 1993.

[6] M. Xiong, J. Wang, and P. Wang, "Differential-algebraic approach to linear programming," *Journal of Optimization Theory and Applications*, vol. 114, no. 2, pp. 443–470, 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1016095904048

[7] L. V. Ferreira, E. Kaszkurewicz, and A. Bhaya, "Synthesis of a k-winners-take-all neural network using linear programming with bounded variables," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3, July 2003, pp. 2360–2365 vol.3.

[8] Y. Zhang and K. Chen, "Global exponential convergence and stability of wang neural network for solving online linear equations," *Electronics Letters*, vol. 44, no. 2, pp. 145–146, January 2008.

[9] Q. Liu and J. Wang, *A One-Layer Dual Recurrent Neural Network with a Heaviside Step Activation Function for Linear Programming with Its Linear Assignment Application*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 253–260.

[10] V. Utkin, *Sliding Modes in Control and Optimization*. Springer Verlag, 1992.

[11] S. K. Korovin and V. I. Utkin, "Using sliding modes in static optimization and nonlinear programming," *Automatica*, vol. 10, no. 5, pp. 525 – 532, 1974.

[12] Q. Liu and J. Wang, *A One-Layer Recurrent Neural Network for Non-smooth Convex Optimization Subject to Linear Equality Constraints*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1003–1010.

[13] A. Bhaya and E. Kaskurewicz, "A control-theoretic approach to the design of zero finding numerical methods," *IEEE Trans Autom Control*, vol. 52, no. 6, pp. 1014–1026, June 2007.

[14] B. W. Wah and T. Wang, "Efficient and adaptive lagrange-multiplier methods for nonlinear continuous global optimization," *Journal of Global Optimization*, vol. 14, no. 1, pp. 1–25, 1999. [Online]. Available: http://dx.doi.org/10.1023/A:1008203422124

[15] X.-S. Zhang, *Neural Networks in Optimization*. Springer, 2010.

[16] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proc. Second Berkeley Symp. on Math. Statist. and Prob. (Univ. of Calif. Press)*, 1951.

[17] A. Polyakov, "Nonlinear feedback design for fixed-time stabilization of linear control systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 8, pp. 2106–2110, 2012.

[18] J. D. Sánchez-Torres, M. J. Loza-Lopez, R. Ruiz-Cruz, E. N. Sanchez, and A. G. Loukianov, "A simple recurrent neural network for solution of linear programming: Application to a microgrid," in *2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*, Dec 2014, pp. 1–7.