Applications of Recurrent Neural Networks to Optimization Problems

Alaeddin Malek
*Department of Applied Mathematics,
Tarbiat Modares University, Tehran,
Iran*

1. Introduction

The work in this chapter presents some applications of recurrent neural networks to general optimization problems. While particular problems presented in this research relates to linear, quadratic and nonlinear programming, monotone variational inequalities and complementarity problems, I fell that the methodology by which one solves these problems are quite general and warrants attention in and of themselves. Correspondingly, I hope that this material will be taken as both a response to a particular problem and a general method.
Constrained optimization problems are defined as the mathematical representation of real world problems concerned with the determination of a minimum or a maximum of a function of several variables, which are required to satisfy a number of constraints. Such function optimization are sought in diverse fields, including mechanical, electrical and industrial engineering, operational research, management sciences, computer sciences, system analysis, economics, medical sciences, manufacturing, social and public planning and image processing.
Although many classical optimization algorithms such as simplex, Karmarkar interior point, direct and indirect techniques are given to solve linear, quadratic and nonlinear optimization problems, in many applications, it is desire to have real-time on-line solutions of corresponding optimization problems. However, traditional optimization algorithms are not suitable for real-time on-line implementation on the computer. The dynamical system approach is one of the promising approaches that can handle these difficulties.
In the recent years many artificial neural networks models developed to solve optimization problems. Several basic and advance questions associated with these models have motivated the studies presented in this chapter.
The goal of this chapter is twofold. The theoretical areas of interest include fundamental methods, models and algorithms for solving general optimization problems using artificial recurrent neural networks. On the other hand, it will try to present and discuss the numerical analysis for the corresponding models, simulations and applications of recurrent neural networks that solve various practical optimization problems.
Recurrent dynamical neural network is an area of neural networks which is one of the fundamental topics of the subject, and combines many mathematical concepts like ordinary and partial differential equations, dynamical systems, unconstrained and constrained

Source: Recurrent Neural Networks, Book edited by: Xiaolin Hu and P. Balasubramaniam, ISBN 978-953-7619-08-4, pp. 400, September 2008, I-Tech, Vienna, Austria

optimization, local and global optima for a function of several variables, sigmoid functions, error estimation, integration and gradient descent methods. Students are often familiar with the local optima of a function with one variable before embarking on an undergraduate course, and in practical way will have integrals which they can not express in closed analytical form. Here we must compute the optimal solution for the constrained optimization problem with objective function of several variables that corresponds with the solution of a system of ordinary differential equations. From mathematical point of the view convergence of the solution and stability of the method has quiet importance, while as an engineer we might look for an algorithm that works for many different problems.

The troublesome problem of just what numerical optimization analysis is arises in recurrent dynamical neural network, as it does in other branches of the field. Should the optimization analysis part be the main aim, or is it the generation of an efficient, tested and validated program which is important? The answer is surely that both areas are important, but at the end of the day numerical analysis and mathematical techniques are some service industry and what the customers want is reliable codes to solve their problems. The theoretical analysis forms part of the reliability assessment, as it determines bounds on errors and levels of stability. These error bounds form the basis of a theoretical justification for the solution convergence of the corresponding numerical algorithm to the actual solution of the original neural network model.

The chapter covers a range of topics from early undergraduate work on constrained linear and quadratic programming through to recent research on nonlinear constrained optimization problems and recurrent neural networks. The source of the optimization work is the lecture notes for graduate students participated in my advance linear programming and optimization courses. The notes have grown in sixteen years of teaching the subject. The work on recurrent neural network models is based partly on my own research. It has taken annual updates as new models have proposed in some of the thesis of my postgraduate students during the last ten years. This research is enriched by the huge literature which has grown in the last two decades.

I am grateful to the applied mathematics department here at Tarbiat Modares University which has made available the technical equipment for the work. The novel models and numerical programs have been tested, compared and improved using the various computers which have been installed over the years.

In the next section we study solution methods for general optimization problems under the assumption that there exists an optimal solution.

2. Optimization problems

In this section, we shall first consider an important class of constrained linear programming problems and their general dual form. Second, we shall introduce primal and dual form of a constrained convex quadratic programming problem. Then we will consider the nonlinear convex programming problems. This, as we shall see, leads to discovering some primal-dual relationships that exists for corresponding class of constrained optimization problems. Among the class of constrained optimization problems, an important and richly studied subclass of problem is that of convex programs.

Definition 1. The problem of maximizing a concave function or minimizing a convex function over a convex set is known as convex programming.

2.1 Constrained linear optimization problems

A problem of the form

$$\text{(PLP)}\quad \begin{array}{ll} \textit{Maximize} & z = c^T x \\ \textit{Subject to} & \\ & Ax \geq b \\ & x \geq 0 \end{array} \tag{1}$$

is said to be a primal linear programming problem, where $x \in R^n, c \in R^n, A \in R^{m\times n}, b \in R^m$. Here $A = (a_{ij})$ is the coefficient matrix of the inequality constraints, $b = (b_1,...,b_m)^T$ is the vector of constants, the components of $c = (c_1,...,c_n)$ are called cost factors, $x = (x_1,...,x_n)^T$ is the vector of variables, called the decision variables. Associated with (PLP) is the linear programming problem (DLP), called the dual of (PLP):

$$\text{(DLP)}\quad \begin{array}{ll} \textit{Minimize} & v = b^T y \\ \textit{Subject to} & \\ & A^T y \leq C \\ & y \geq 0 \end{array} \tag{2}$$

In (DLP) formulation y is the vector of m dual variables. We can define the dual of any linear problem after writing it in the primal form (PLP), [1].

Remark 1. Primal and dual linear programs (PLP) and (DLP) are convex programs since the set of feasible solutions to a linear program is a convex set and a linear objective function is both convex and concave.

2.2 Constrained quadratic optimization problems

We consider a primal quadratic programming problem in

$$\textit{Minimize} \quad f(x) = \frac{1}{2}x^T Ax + c^T x$$

$$\textit{subject to} \quad g(x) = Dx - b = 0, \quad x \geq 0, \tag{3}$$

Where A is a $m \times m$ symmetric positive semidefinite matrix, D is a $n \times m$ matrix and rank (D) = m, $b \in R^n$, $x, c \in R^m$. We define the dual problem (DQP) as follows:

$$\textit{Minimize} \quad \tilde{f}(x) = -\frac{1}{2}x^T Ax + b^T y$$

$$\textit{Subject to} \quad \tilde{g}(x) = D^T y - \nabla f(x) \leq 0, \tag{4}$$

where $\nabla f(x) = Ax + c,\ y \in R^n$.

Lemma 1. The primal quadratic program (PLP) and its dual (DLP) are convex programs. This is because the quadratic forms $\frac{1}{2}x^T Ax + c^T x$ and $-\frac{1}{2}x^T Ax + b^T y$ are convex if and only if A is a positive semidefinite matrix (for example see [2]). Clearly the standard linear programming problem

$$\begin{aligned} &Maximize \quad z = c^T x \\ &Subject\ to \\ &\qquad Dx = b \\ &\qquad x \geq 0 \end{aligned} \tag{5}$$

and its dual

$$\begin{aligned} &Minimize \quad v = b^T y \\ &Subject\ to \\ &\qquad A^T y \geq c \\ &\qquad y\ is\ free\ in\ sign, \end{aligned} \tag{6}$$

are special cases of the (3) and (4) respectively, for which $A = 0_{m\times m}$.

2.3 Constrained nonlinear optimization problems

Consider the following nonlinear convex programming problem (NP) with nonlinear constraints:

$$\text{(NP)}\ Minimize \quad f(x)$$

$$Subject\ to\ g(x) \leq 0,\ x \in \Omega \tag{7}$$

where $x = (x_1, \ldots, x_n)^T \in R^n$, $f : R \to R^n$. $g(x) = (g_1(x), \ldots, g_m(x))$ is m-dimensional vector-valued continuous function of *n* variables. The functions f and $g_1, \ldots, g_m$ assumed to be convex and twice differentiable for $\Omega \subseteq R^n$.

Definition 2. A vector x is called a feasible solution to (NP) if and only if *x* satisfies $m+n$ constraints of the (NP).

Definition 3. Any feasible solution x is said to be a regular point if the gradients of $g_i(x)$, $\nabla g_i(x)$ for $(i \in I = \{j \mid g_j(x) = 0\})$, are linearly independent.

Definition 4. The (NP) has at least one optimal solution [3] when

i. the set of all feasible solutions is nonempty and bounded,

ii. the feasible set is unbounded but $f(x)$ has a bound level set.

2.4 Monotone variational inequalities and complementarity problems

The problem of finding a vector point $x^* \in R^n$ such that

$$x^* \in S, \quad \langle F(x^*),\ x - x^* \rangle \geq 0 \quad \text{for all } x \in S \tag{8}$$

where $x = (x_1, \ldots, x_n)^T \in R^n$, is called the monotone variational inequality problem [4]. F is a continuous mapping from R^n into itself, and $S = \{x \in R^n \mid Ax - b \geq 0,\ Bx = c,\ x \geq 0\}$ where $A \in R^{m \times n}$, rank (A)= m, $B \in R^{r \times n}$, rank(B)= r, $0 \leq m$, $r \leq n$, $b \in R^m$, $c \in R^r$, and S is a nonempty closed convex subset of R^n and $\langle\ .\ ,\ .\ \rangle$ denotes the inner product in R^n. In the special case where $S = R^n_+$, problem (8) can be rewritten as the following nonlinear complementarity problem

$$x^* \geq 0, \quad F(x^*) \geq 0, \ \langle x^*,\ F(x^*) \rangle = 0\,. \tag{9}$$

For $S = R^n$, problem (8) reduces to solving the system of nonlinear equation $F(x) = 0$, [5].

Remark 2. For a continuously differentiable function f, if x^* is a solution of the problem Minimize$\{f(x) \mid x \in S\}$; $S = \{x \in R^n \mid Ax - b \geq 0,\ Bx = c,\ x \geq 0\}$ then x^* is also a solution of (8) with $F(x) = \nabla f(x)$, and $\nabla f(x) = \left(\partial f / \partial x_1, \ldots, \partial f / \partial x_n\right)^T \in R^n$ is the gradient vector of $f(x)$ at point x.

Definition 5. [6] A mapping $F : R^n \to R^n$ is said to be monotone on S if

$$\langle F(x) - F(x'),\, x - x' \rangle \geq 0 \quad \text{for all } x, x' \in S. \tag{10}$$

F is strictly monotone on S, if strict inequality holds in (10) whenever $x \neq x'$.

Lemma 2. If F is continuously differentiable and the Jacobean matrix ∇F is positive definite for all $x \in S$, i.e.

$$\langle d, \nabla F d \rangle > 0 \quad \text{for all } \ x \in S,\ d \in R^n \ (d \neq 0)\,.$$

then F is strictly monotone on S.

Proof. For example see [7].

The variational inequalities problems have wide variety of scientific and engineering applications (for example see [2], [6], [8] to [11]). In many applications, real-time on-line solutions of (8) and (9) are desired. However, traditional algorithms (see [2], [6], [8], [12] and [13]) are not suitable for real-time on-line implementation on the computer. One promising approach to handle these problems is to employ an artificial neural network based on circuit implementation. Many continuous-time neural networks for constrained optimization problems, have been developed ([14] to [18]) using network parameters. To avoid using penalty parameters, some significant works have been done in recent years. A few primal and dual neural networks with two-layer and one-layer structure were developed in [14], [17] and [18]. These neural networks were proved to be globally convergent to an exact solution.

In the next section, we discuss some general ideas about artificial neural networks.

3. Artificial neural networks

Artificial neural networks consist of a calculation unit called neuron. Every neuron has some real valued inputs. Inside every neuron, each input is multiplied with corresponding neural coefficient defining its value. The sum of all these products adds to a value called bias. Finally, activation function affects this sum and determines the real valued output of the neuron feed forwardly [19] or by some feed back [20].

3.1 Feed forward back propagation neural networks

Primary discussions regarding artificial neural networks introduced in the 40's with presentation of the feed forward neural networks. Artificial neural networks in some extents are modeled from the brain and neural system of the human, which are able to give acceptable solutions based on correct information records from the problem.
The basic structure for the feed forward back propagating neural network (nets without feed back) consists of some number of nodes in the input layer, the hidden layer, and the output layer that has one node. The sigmoid functions approximate linear functions, yet allow the update scheme to propagate backwards through differentiable functions. The manner in which input data generates output data for a given neural network depends on the interconnection weights. These weights are adjusted to reduce the error between the neural network outputs and the actual output values. i.e.

$$E = \frac{1}{2}\sum_{i=1}^{n}(O_i^{actual} - O_i^{net}) \tag{11}$$

where O_i^{actual} is the actual output for the i^{th} training point. O_i^{net} is the estimated value from the neural network for the i^{th} training point from the neural network. n is the total number of training points obtained by taking known data points for a given task. Here the objective is to train the network so that the output from the network minimizes equation (11).

3.2 Recurrent dynamical artificial neural network

Khanna in year 1990 [21], describes associative memory as "the ability to get from one internal representation to another or to infer a complex representation from a portion of it". Effectively our goal in applying neural networks is to create a functional mapping from steady optimization space to either dynamical time dependent space or some parameter space. Two approaches to achieving this mapping have been extensively studied by Xia [14], [15] and [22] to [25], Malek [4], [16], [26] and [27] and their coauthors.
The first approach relies on a structure with adjustable parameters. On the basis of known input/output pairs, these parameters are selected or changed. If this approach is successful, the appropriate selection of these parameters will yield a mapping device which will always provide the associated output values for a given input.
The second approach uses information from the primal and dual optimization problem and applied primarily by Malek in year 2005 [16]. The basis for such systems is a precisely defined set of ordinary differential equations that automatically satisfy the related primal and dual optimization problems simultaneously. These information are defined by the cumulative designing the system and are laid out in a hierarchical fashion. The system then

performs a sequential set of values, using the output from the previous as the input to the next. If successful, a system can be created which will associate input with its correlated output. The challenge is to make the system complete enough (consistent, convergent and stable) to always associate the correct output with a given input.
The primary difference in these two approaches is that adjustable parameters in the first are a prior, i.e., the parameters are settled upon and maintained before data is introduced into the system. The second approach has no adjustable parameters thus its model is simple to use. The advantage of this approach is that in this way, we can obtain a solution for the given real life problem, however we wish to assume a prior knowledge of relationships between constrained optimization problem and dynamical system. Moreover the solution for optimization problem consists of a solution for real life problem, since optimization problem is simulated from the corresponding real life problem.
The work presented in this section applies recurrent dynamical artificial neural network. We shall emphasize on networks that do not use network parameters or penalty parameters in advance. This approach is a metric driven method. i.e., we establish distance between the input and the neural network output. For a given input, the neural network outputs the value whose distance from the given input is smallest using linear constraint least square technique or any other related method. One manner of doing this mapping is to associate the equilibrium points of a dynamical system with the optimal points of constraint optimization problem. When the input is the initial condition of the dynamical system, the system will converge to an equilibrium point. Thus this optimal solution contains a solution that minimizes equation (11), where we use the feed back process to produce corresponding optimal weights. This means that the artificial neural network structure is recurrent.
The structure of the recurrent dynamical artificial neural network is different from the feed forward artificial neural network. However it is possible to make some corresponding relations between these two neural networks (see Rumelhart 1986, [28]). i.e., there is a sense in which the error back propagation scheme may be applied to networks that contain feed back, (see Fig. 3.1). The feed forward network in Fig. 3.1 may be represented to simulate a feed back network with a given set of weight and bias parameters.
Having developed the equivalent structure as shown in Fig. 3.2, it becomes proper to say "the goal for recurrent dynamical artificial neural network, as with the back propagation artificial neural network, is to minimize the error function given by equation (11).
Training of dynamical neural networks has received considerable attention in the last 30 years [20], [29] and [30]. The equations governing the behavior of the simplest supervised recurrent dynamical neural network are

$$\frac{\partial u}{\partial t} = -u + AS(u) + Bx, \qquad u_{initial} = 0 \tag{12}$$

$$u^* = AS(u^*) + Bx \tag{13}$$

$$y = C^T u^* \tag{14}$$

where $$S(u) = \frac{1}{1+e^{-u}}$$

is the sigmoid function. The adjustable parameters in this supervised recurrent dynamical neural network are found in the *A*, *B* matrices and vector *C*. The input x of the neural network corresponds to the input data associated with a training point. This input is then applied to the system governed by equation (12). When equation (13) reaches an equilibrium value *u** for this input, we obtain the output of the neural network by taking the dot product of *C* and *u** by equation (14). This neural network output will then compare with the actual output. To update the elements of *A*, *B*, and *C*one may use gradient descent method using $\frac{\partial u}{\partial A_{ij}}, \frac{\partial u}{\partial B_{ij}}$, and $\frac{\partial u}{\partial C_i}$.

This minimization task requires that the neural network possess enough parameter freedom to enable each input set to generate an output close to the actual value. This is not a case in many problems. Thus in the next section we emphasize on the unsupervised recurrent dynamical artificial neural networks.

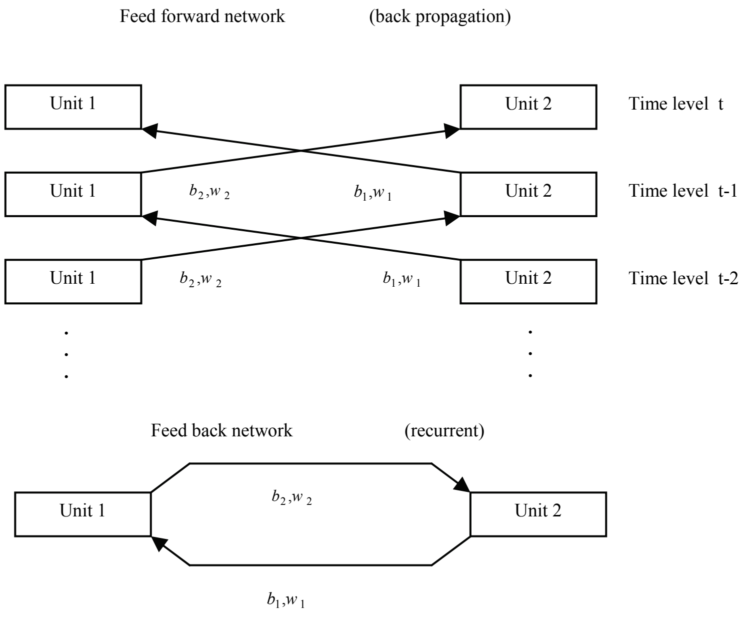

Fig. 3.1 Equivalent structures of a two unit network; Feed forward network, and feed back network for a given biases b_1 and b_2 and weights w_1 and w_2.

4. Networks dynamic analysis

For many times dependent cost functions an online optimizer on the basis of an analog circuit [31], [32] and [33]) is desirable. Dynamic solvers or analog computer, was first proposed by Dennis [34], Rybashov [35] and [36], Karpinskaya [37], and later studied by Kenedy and Chua [38], Rodriguez-Vazquez et al. [39], Tank and Hopfield [31]. These dynamic solvers usually employ neural networks since they have many advantages over the traditional algorithms. Massively parallel processing and fast convergence are two of the most important advantages of the neural networks.

4.1 Models for linear programming

Use of neural network for the solution of linear programming problems goes back to 1985, when Hopfield and Tank [31] provide fast algorithm based on analog electrical components. Chen and Fang [40] in 1998 examined the theoretical properties of a method proposed by Kennedy and Chua in 1987, [38]. Malek and Yari in year 2005 proposed a fully stable artificial recurrent neural network model for the solution of primal linear programming problems of the type (1):

$$\begin{cases} \dfrac{dX(t+\eta)}{dt} = \eta_1[C - A^T(Y + \eta\dfrac{dY}{dt})] \\ \dfrac{dY(t+\eta)}{dt} = \eta_2[A(X + \eta\dfrac{dX}{dt}) - b] \end{cases} \tag{15}$$

where η, η_1 and η_2 are rate of learning (in the neural network dynamic). They are step sizes in the process of optimization computation. η, η_1 and η_2 can stay constant or vary in each iteration.

Model (15) transfers the linear programming problem into a dynamical system of equations and gives approximation solution to the exact solution only for primal variables. This means that by the recurrent neural network model (15) dual optimum value for objective function does not coincide exactly with the optimum value obtained from primal problem.

The second model proposed by Malek in the same article is in the following form [16]:

$$\begin{cases} \dfrac{dU(t+\eta)}{dt} = \eta_1[\overline{C} - \overline{A}^T(V + \eta\dfrac{dV}{dt})] \\ \dfrac{dV(t+\eta)}{dt} = \eta_2[\overline{A}^T(U + \eta\dfrac{dU}{dt}) - \overline{b}] \end{cases} \tag{16}$$

where $U = (X, Y)$ and V is the corresponding dual variable to the dual form of problem

$$\begin{aligned} &Maximize \quad Z = \left[C^T(X + \eta\frac{dX}{dt}) - b^T(Y + \eta\frac{dY}{dt}) \right] \\ &Subject\ to \quad A(X + \eta\frac{dX}{dt}) \le b \\ &\qquad\qquad -C^T\frac{dX}{dt} \le 0 \\ &\qquad\qquad -A^T(Y + \eta\frac{dY}{dt}) \le -C \\ &\qquad\qquad b^T\frac{dY}{dt} \le 0 \\ &\qquad\qquad X + \frac{dX}{dt} \ge 0, \ Y + \eta\frac{dY}{dt} \ge 0 \end{aligned} \tag{17}$$

$\overline{A}$ is a block matrix of the form

$$\overline{A} = \begin{pmatrix} A & \eta A & 0 & 0 \\ 0 & -C^T & 0 & 0 \\ 0 & 0 & -A^T & -\eta A^T \\ 0 & 0 & 0 & b^T \end{pmatrix}$$

for $A_{m\times n}$, $\overline{C} = (C^T, \eta c^T, -b^T, -\eta b^T)$, $\overline{b} = (b, 0, -c, 0)$. We shall see that, $\overline{A}$ is a *(m+n+2)* × $(2m+2n)$ matrix and $\overline{C}$ is a vector with $2m+2n$ components and $\overline{b}$ is$(m+n+2)\times 1$ vector.

The following lemma shows that this model solves both primal and dual problems of the type (1) and (2) simultaneously.

Lemma 3. For $X^* = (x_1^*, x_2^*, ..., x_n^*)$ the optimum solution $U^* = (X^*, Y^*)$ of problems in the forms (PLP) and (DLP), is the optimum solution for (P-D) *iff* Z^* the maximum value for Z vanishes where $\frac{dX}{dt} \to 0$ and $\frac{dY}{dt} \to 0$.

Proof: See [16].

These models need some network parameters η, η_1 and η_2 that must be fixed in the starting time.

4.2 Models for quadratic programming

Xin-Yu Wu et al. [22] in year 1996 proposed the following neural network model to solve problems (3) and (4)

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\begin{Bmatrix} \beta(-D^T y + Ax + c) + \beta A[x - (x + D^T y - Ax - c)^+] + D^T(Dx - b) \\ \beta\{Dx - b + D[(x + D^T y - Ax - c)^+ - x]\} \end{Bmatrix} \tag{18}$$

where $\beta = \left\| x - (x + D^T y - Ax - c)^+ \right\|_2^2$.

Youshen Xia [14] considered the adjusted form of model (1) as follows

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\begin{Bmatrix} (I + A)[x - (x + D^T y - Ax - c)^+] + D^T(Dx - b) \\ -D[x - (x + D^T y - Ax - c)^+] + Dx - b \end{Bmatrix} \tag{19}$$

where I is the identity matrix.

Malek and Oskoei [26] proposed three novel models based on model (1) in the following forms:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{Bmatrix} D^T y - Ax - c - A[x - (x + D^T y - Ax - c)^+] - D^T(Dx - b) \\ -D(x + D^T y - Ax - c)^+ + b \end{Bmatrix} \tag{20}$$

Model (20) is a simplified model (18) of Xin-Yu Wu et al. Here one may concerne of obtaining better accuracy for the final solutions, while we do not use expensive analog multipliers of Xin-Yu Wu et al. Therefore the relative question might be: is there a simpler neural network models in the manipulation of hardware tools. Malek & Oskoei [26] show that for some examples model (20) converges to the exact solution with 13 exact decimal points. While in the same conditions the solutions for neural network proposed by Xin-Yu Wu agrees with the corresponding exact solution only up to 3 decimal points.
It is still possible to simplify model (20). Model (21) has the advantage of serious simplification and good accuracy in the same time. It is in the form [26]:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\begin{Bmatrix} (I+A)[x-(x+D^T y - Ax - c)^+] \\ D(x+D^T y - Ax - c)^+ - b \end{Bmatrix} \tag{21}$$

Let us assume that

$$(x, y) \in \Psi,\ \Psi = \left\{(x,\ y)\middle| y \in R^n, x \in R^m, x \ge 0\right\},$$

$(x)^+ = \left[(x_1)^+, \dots, (x_m)^+\right]^T$ and $(x_i)^+ = \max\{0, x_i\}$, for $i = 1,\dots,m.$ We proposed following model:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = -\begin{Bmatrix} x-(x+D^T y - Ax - c)^+ \\ D[(x+D^T y - Ax - c)^+]-b \end{Bmatrix} \tag{22}$$

in [26] which appears to be more efficient than the models (20) and (21) when we investigate the complexity, complexity of individual neurons, stability, and accuracy of the solutions, (see Tables 1 and 2 in section 5).
Model (22) does not use any projection operator in practice thus it is different and simpler from the model proposed by Qing Tao et al. Here in model (9), unlike the Qing Tao's model we do not use any extension of Newton's optimal descent flow equation to solve the problem.
If we assume that $\alpha = (x + D^T y - Ax - c)^+$ and $\beta = D^T(Dx - b)$, then models (22) and (19) are in the following forms respectively [41]:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{cases} \alpha - x, \\ \\ -D\alpha + b. \end{cases} \tag{23}$$

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{cases} (I+A)\ (\alpha - x) - \beta, \\ \\ -D\alpha + b. \end{cases} \tag{24}$$

The network circuit implementation for solving problems (3) and (4) whose dynamics are governed by (23) are given in the Fig. 3.2. The circuit consists of adders (summing amplifiers) and integrators. In the Fig. 3.2, vectors c and b are external input vectors, while

x and y are the network outputs. In this diagram dynamical process of vector α is the same as what is given in [14]. A simplified block diagram of α is illustrated in Fig. 3.3 to show how expensive it is using vector α in the arbitrary model.
Malek & Alipour, Applied Mathematics and Computation 192 (2007) 27-39 We now compare the network (24) with our proposed network in (23) for solving problems (2) and (3). The network (24) is stable to exact solution and there are no parameters to set, but the main disadvantage of it is that too many expensive analog multipliers (α, β) are required for large scale quadratic programming problems, thus the set of hardware implementation is expensive and therefore greatly affect the accuracy of solutions. Neural network model (23) does not need to use β and therefore in practice needs relatively less computational efforts. Moreover, this model is globally convergence to the corresponding exact solution independent of where and how to choose the starting input initial values. Model (23) not only has the same global convergence property as the model (24), but also has some more advantages, plus simplicity. Network (23) is better than network (24) in the sense of complexity, i.e. usage analog multipliers and hardware implementations.

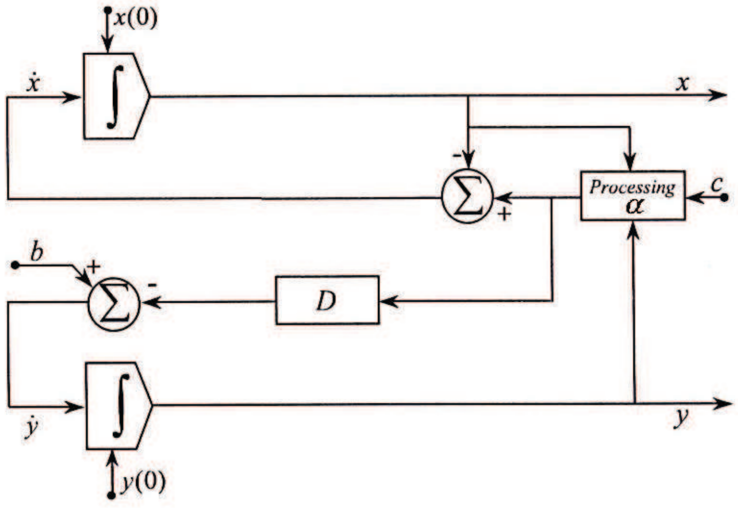

Fig. 3.2. A simplified neural network diagram for model (23): Malek & Alipour, Applied Mathematics and Computation 192 (2007) 27-39

Remark 3. Model in (23) may be used for solving general standard linear programming problems by setting $A = 0_{m\times m}$.
Simulation and numerical results are discussed in the next section.
Theorem 1. The recurrent dynamic artificial neural network (23) is globally convergent to the solution set of the primal and dual quadratic programming problems (3) and (4).
Proof. Let in the proposed model of Qing Tao et al. [17], general projection operator to be the identity operator. Then the proof is similar to Qing Tao's proof. (see [26] and also see Theorem 4)
In the reminder of this subsection we will try to clarify the ideas in Theorem 1 from theoretical point of view (see [41]).

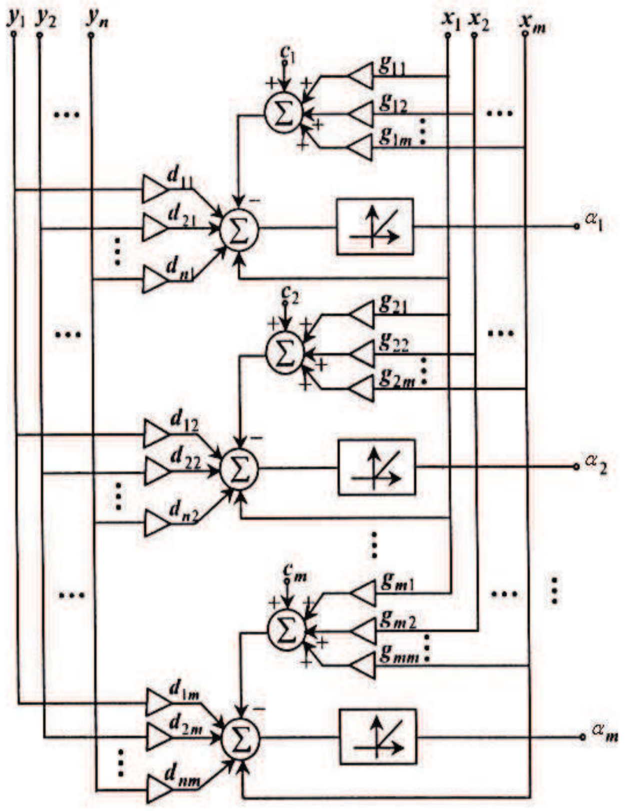

Fig. 3.3. A simplified block diagram for α , where $A = (a_{ij})$ and $D = (d_{ij})$:

In this section, we shall study the dynamics of network (23).We define a specific Liapunov function and get the global convergence of network (23).We first discuss some prerequisites.

Definition 6. A continuous-time neural network is said to be globally convergent if for any given initial point, the trajectory of the dynamic system converges to an equilibrium point.

Lemma 4. Let Ψ be a closed convex set of R^m .Then

$$\left[v - p_\Psi(v)\right]^T \left[p_\Psi(v) - x\right] \geq 0, v \in R^m, x \in \Psi$$

and $\left\|p_\Psi(v) - p_\Psi(u)\right\| \leq \|v - u\|, v, u \in R^m$

where $\|.\|$ denote l_2 norm and the projection operator $p_\Psi(u)$ is defined by

$$p_\Psi(u) = \arg\min_{v\in\Psi} \|u - v\|.$$ Proof. See [42].

Remark 4. Since $R_+^m = \{x | \ x \geq 0\}$ is a closed convex and by the property of a projection on a

$$\left[v - (v)^+\right]^T \left[(v)^+ - \tilde{x}\right] \geq 0, \tilde{x} \in R_+^m, v \in R^m.$$

Theorem 2. x^*, y^* are solutions of problems (3) and (4), respectively, if and only if (x^*, y^*) satisfies

$$\begin{cases} (x^* + D^T y^* - Ax^* - c)^+ = x^*, \\ D(x^* + D^T y - Ax^* - c)^+ = b. \end{cases} \tag{25}$$

Proof . By Karush-Kuhn-Tucker theorem for convex programming problem [43] we have x^*, y^* are solutions of problems (3) and (4), respectively, if and only if (x^*, y^*) satisfies

$$\begin{cases} Dx^* = b, x^* \geq 0, \\ x^{*T}(D^T y^* - Ax^* - c) = 0, \\ D^T y^* - Ax^* - c \leq 0. \end{cases} \tag{26}$$

Clearly, that (26) is equivalent to (25).
We will now prove a theorem that is a base for proving the global convergence of model (23).

Theorem 3. Let $F_1(x,y) = \frac{1}{2}(x - x^*)^T A(x - x^*) + \frac{1}{2}\|x - x^*\|^2$ and $F_2(x,y) = \frac{1}{2}\|y - y^*\|^2$ and $F(x,y) = F_1(x,y) + F_2(x,y)$. Then

$$\frac{d}{dt}F(x,y) \leq -(x - x^*)^T A(x - x^*) - \left\|x - \left(x + D^T y - Ax - c\right)^+\right\|^2.$$

Proof .

$$\frac{d}{dt}F_1(x,y) = \left(A(x - x^*)\right)^T \frac{dx}{dt} + \left(x - x^*\right)^T \frac{dx}{dt}$$

$$= \left(A(x - x^*)\right)^T \left((x + D^T y - Ax - c)^+ - x\right) + \left(x - x^*\right)^T \left((x + D^T y - Ax - c)^+ - x\right)$$

Note that

$$\left(A(x - x^*)\right)^T \left((x + D^T y - Ax - c)^+ - x\right)$$

$$= \left(A(x - x^*)\right)^T \left((x + D^T y - Ax - c)^+ - x^* + x^* - x\right)$$

$$= \left(A(x - x^*)\right)^T \left(x^* - x\right) + \left((x + D^T y - Ax - c)^+ - x^*\right)^T \left(Ax + c\right)$$
$$- \left((x + D^T y - Ax - c)^+ - x^*\right)^T \left(Ax^* + c\right)$$

On the other hand,

$$
\begin{aligned}
&\left((x+D^T y-Ax-c)^+-x^*\right)^T (Ax+c)\\
&=\left((x+D^T y-Ax-c)^+-x^*\right)^T \left(Ax+c-D^T y-x+(x+D^T y-Ax-c)^+\right)\\
&+\left((x+D^T y-Ax-c)^+-x^*\right)^T \left(x-(x+D^T y-Ax-c)^+\right)\\
&+\left((x+D^T y-Ax-c)^+-x^*\right)^T D^T y\\
&=\left((x+D^T y-Ax-c)^+-x^*\right)^T \left(Ax+c-D^T y-x+(x+D^T y-Ax-c)^+\right)\\
&+\left((x+D^T y-Ax-c)^+-x\right)^T \left(x-(x+D^T y-Ax-c)^+\right)\\
&+\left(x-x^*\right)^T \left(x-(x+D^T y-Ax-c)^+\right)\\
&+\left((x+D^T y-Ax-c)^+-x^*\right)^T D^T y
\end{aligned}
$$

and

$$\left((x+D^T y-Ax-c)^T-x^*\right)^T (Ax^*+c)$$

$$=\left((x+D^T y-Ax-c)^T-x^*\right)^T \left(Ax^*+c-D^T y^*\right)+\left((x+D^T y-Ax-c)^T-x^*\right)^T D^T y^*$$

So

$$
\begin{aligned}
&\frac{d}{dt}F_1(x,y)=\left(A(x-x^*\right)^T\left(x-x^*\right)\\
&+\left((x+D^T y-Ax-c)^+-x^*\right)^T \left(Ax+c-D^T y-x+(x+D^T y-Ax-c)^+\right)\\
&-\left\|x-(x+D^T y-Ax-c)^+\right\|^2-\left((x+D^T y-Ax-c)^+-x^*\right)^T\left(Ax^*+c-D^T y^*\right)\\
&+\left((x+D^T y-Ax-c)^+-x^*\right)^T\left(D^T y-D^T y^*\right).
\end{aligned}
$$

Thus by (22) we have

$$
\begin{aligned}
&\left((x+D^T y-Ax-c)^+-x^*\right)^T\left(D^T y^*-Ax^*-c\right)\\
&=\left((x+D^T y-Ax-c)^+\right)^T\left(D^T y^*-Ax^*-c\right)-x^{*T}\left(D^T y^*-Ax^*-c\right)\\
&=\left((x+D^T y-Ax-c)^+\right)^T\left(D^T y^*-Ax^*-c\right)\le 0
\end{aligned}
$$

Using lemma 4 we have

$$
\begin{aligned}
\frac{d}{dt}F_1(x,y)\le&-\left(x-x^*\right)^T A\left(x-x^*\right)-\left\|x-(x+D^T y-Ax-c)^+\right\|^2\\
&+\left((x+D^T y-Ax-c)^+-x^*\right)^T\left(D^T y-D^T y^*\right).
\end{aligned}
$$

Since

$$\frac{d}{dt}F_2(x,y) = \left(y-y^*\right)^T \frac{dy}{dt}$$

$$= \left(y-y^*\right)^T \left(-D(x+D^Ty-Ax-c)^+ + Dx^*\right)$$

then

$$\frac{d}{dt}F(x,y) \le -\left(x-x^*\right)^T A\left(x-x^*\right) - \left\| x-\left(x+D^Ty-Ax-c\right)^+ \right\|^2 .$$

The proof is complete.

Theorem 4. Network (4) is globally convergent to the solutions set of problems (3) and (4).

Proof . Using lemma 4, the right hand side of (23) is a Lipschitz mapping. From the existence theory of ordinary differential equations [44], we can assume that for any $(x_0, y_0) \in R^m \times R^n$ there exists a unique solution $\left(x(t), y(t)\right)$ of (4) and its maximal existence interval $\left[0, \lambda(x_0, y_0)\right)$.

Let x^*, y^* be solutions of problems (3) and (4) respectively. Let

$$V = \left\{ (x,y) \in R^m \times R^n \,\middle|\, F(x,y) \le \frac{1}{2}(x_0 - x^*)^T A(x_0 - x^*) + \frac{1}{2}\left\|x-x^*\right\|^2 + \frac{1}{2}\left\|y-y^*\right\|^2 \right\}$$

Using theorem 3, $F(x,y)$ is a Liapunov function of system (23) on *V*. Since $(x-x^*)^T A(x-x^*) \ge 0$ we have

$$F(x,y) \ge \frac{1}{2}\left\|x-x^*\right\|^2 + \frac{1}{2}\left\|y-y^*\right\|^2 .$$

This proves that *V* is bounded. By the extension theory of ordinary differential equations [], $\lambda(x_0, y_0) = +\infty$ .Using the LaSalle invariant principle [45], there exists a constant k, such that $\left(x(t), y(t)\right) \to M \cap F^{-1}(k), t \to +\infty$, where *M* is the maximal invariant set in

$$\Omega = \left\{ (x,y) \,\middle|\, \frac{d}{dt}F(x,y) = 0, (x,y) \in \bar{V} \right\}.$$

Now we will prove that every point in set *M* is a solution of problems (3) and (4). $\forall (x_1, y_1) \in N$, let $\left(x_1(t), y_1(t)\right)$ be a solution of equation (23) with initial point (x_1, y_1), its maximal existence interval is $\left[0, \lambda(x_1, y_1)\right)$. By the invariant of *M* and bounded ness of *V*, we have $\lambda(x_1, y_1) = +\infty$, $x_1(t) = x_1$. If (x_1, y_1) is not a solution of problems (3) and (4), using theorem 2 and 3 $D\left(x_1 + D^T y_1 - Ax_1 - c\right)^+ \ne b$. From (23)

We have $\left\|y_1(t)\right\| \to \infty$ as $t \to \infty$. It is contradictory to the bound ness of *V*. Thus (x_1, y_1) is a solution of problems (3) and (4). Since (x_1, y_1) is arbitrary the proof is completed.

4.3 Models for nonlinear programming

Malek and Yashtini proposed the following recurrent dynamical artificial neural network [46]

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} P_\Omega[x - \nabla f(x) - \nabla g(x)y] - x \\ [y + g(x)]^+ - y \end{pmatrix}, \tag{27}$$

for the solution of nonlinear programming problem:

$$\begin{aligned} &Minimize \quad f(x) \\ &Subject\ \ to \quad Ax \le b \\ &x \in \Omega \end{aligned} \tag{28}$$

where $A \in R^{m\times n}$, $b \in R^m$.

4.4 Models for variational inequalities

The systems governing the behavior of the recurrent dynamical artificial network corresponding to the variational inequalities problem (8) are [4]

$$\frac{du}{dt} = \frac{d}{dt}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} (x - F(x) + A^T y + B^T z)^+ - x \\ (y - Ax + b)^+ - y \\ -Bx + c \end{pmatrix} \tag{29}$$

$$\begin{cases} x^* = (x^* - F(x^*) + A^T y^* + B^T z^*)^+ \\ y^* = (y^* - Ax^* + b)^+ \\ Bx^* = c \end{cases} \tag{30}$$

where $(x_i)^+ = \max\{0, x_i\}$ for all $i = 1, \ldots, n$ and $(y_j)^+ = \max\{0, y_j\}$ for all $j = 1, \ldots, m$, and x^* is the solution of monotone variational inequalities problem (…).

Now, let $x(.)$, $y(.)$ and $z(.)$ be some dependent variables to time t. We initiate $u_{initial} = 0$ to the system governed by (29), when system (30) reaches an equilibrium value u^* for this input, we obtain the output of the neural network. The goal for the continuous time based dynamical system described by two systems (29) and (30), is to minimize the error function given by equation (11).

Yashtini and Malek [4] proved that the recurrent neural network based on the systems (29) and (30) are stable in the sense of Lyapunov and globally convergent to an optimal solution.

5. Work examples

For the following three models proposed by Xia, Malek and their coauthors solve quadratic programming problem in Example 1.