Australian Journal of Basic and Applied Sciences, 5(10): 814-823, 2011
ISSN 1991-8178

A Novel Recurrent Neural Network Model For Solving Nonlinear Programming Problems With General Constraints

Mohsen Alipour

Member of Young Research Club, Islamic Azad University, Sari Branch,
P.O. Box 48164-194, Sari, Iran.

Abstract: In this paper, we propose a recurrent neural network for solving nonlinear convex programming problems subject to linear equality and inequality constraints. Using this network we can solve primal programming problems and their duals, simultaneously. The proposed neural network has a simpler structure than the existing neural networks for solving such problems, and converges very faster to exact primal and dual solution.

Key words: Nonlinear programming problem, Neural networks, Dynamical systems.

INTRODUCTION

The dynamical systems approach to solving constrained optimization problems was first proposed by Pyne (1956), and later studied by Rybashow (1965), Karpinskaya (1967) and others recently, due to renewed interest in neural networks.

In many applications, real-time solutions are usually imperative. One example of such applications in image processing is the solution to the image fusion problem in real-time wireless image transmission see Li *et al*. (1995). Compared with traditional numerical methods for constrained optimization, the neural network approach has several advantages in real-time applications. First, the structure of a neural network can be implemented effectively using VLSI and optical technologies. Second, neural networks can solve many optimization problems with time-varying parameters. Third, the ordinary differential equation techniques can be applied directly to the continuous-time neural networks numerically for solving constrained optimization problems effectively. Therefore, neural network methods for the solution of optimization problems have been received considerable attention see Kennedy *et al*. (1988), Maa *et al*. (1992), Cichocki *et al*. (1993), Zak *et al*. (1995) and Xia (1996). For the primal-dual solution of linear, quadratic and nonlinear programming problems see Malek's models in Malek *et al*. (2005, 2007), Yashtini *et al*. (2007, 2008) and Oskoei *et al*. (2007).

The reminder of this paper is arranged as follows. In Section 2, we introduce the basic problem formulation. In Section 3, a neural network model for solving nonlinear convex programming problems subject to linear equality and inequality constraints is proposed. In Section 4, several examples are considered to evaluate the power and effectiveness of proposed neural network approach. Some conclusions are summarized in the last section.

2. The Basic Problem Formulation:

Consider the following nonlinear convex programming problem subject to linear equality and inequality constraints. We call it the primal problem:

$$\begin{aligned} &Minimize \quad f(x) \\ &subject\ to \quad Dx = b, \\ &\qquad\quad Ax \geq c, \\ &\qquad\quad x \geq 0, \end{aligned} \tag{1}$$

where x is the *n*-dimensional decision vector, $f : R^n \to R$ is a continuously differentiable and convex function on R^n. $D \in R^{m \times n}$, $A \in R^{p \times n}$ are coefficient matrices and $b \in R^m$, $c \in R^p$ are constant column vectors. The minimization problem for the primal problem (1) will be written as

$$Minimize\ L(x, y, z) = f(x) - y^T (Dx - b) - z^T (Ax - c), \tag{2}$$

Corresponding Author: Mohsen Alipour, Member of young research club, Islamic Azad University, Sari branch, P.O. Box 48164-194, Sari, Iran.
E-mail: m.alipour2323@gmail.com

where $L(x, y, z)$ is the Lagrange function and $z \in R_+^p = \{z \in R^p \mid z \geq 0\}$, $y \in R^m$ are Lagrange multipliers. According to the Karush-Kuhn-Tucker (KKT) condition see Bertsekas (1989), x^* is a solution to (1) if and only if there exist $y^* \in R^m, z^* \in R_+^p$ such that (x^*, y^*, z^*) satisfies the following conditions:

$$\begin{aligned}
&\nabla f(x^*) - D^T y^* - A^T z^* \geq 0,\\
&x^{*T}\left(\nabla f(x^*) - D^T y^* - A^T z^*\right) = 0,\\
&b - Dx^* = 0,\\
&c - Ax^* \leq 0,\\
&z^{*T}(c - Ax^*) = 0,
\end{aligned} \tag{3}$$

where ∇f is the gradient of f.

3. Neural network model:

Here, we propose a novel recurrent neural network model for solving problem (1) and its dual as follows:

$$\frac{dx}{dt} = -\nabla f\left(x + k\frac{dx}{dt}\right) + D^T\left(y + k\frac{dy}{dt}\right) + A^T\left(z + k\frac{dz}{dt}\right), \quad x \geq 0, \tag{4}$$

$$\frac{dy}{dt} = b - D\left(x + k\frac{dx}{dt}\right), \tag{5}$$

$$\frac{dz}{dt} = -A\left(x + k\frac{dx}{dt}\right) + c, \quad z \geq 0, \tag{6}$$

where coefficient k is some positive constant. The main property of the above system is stated in following theorem. A block diagram of neural network model $(4)-(6)$ is shown in Fig. 1.

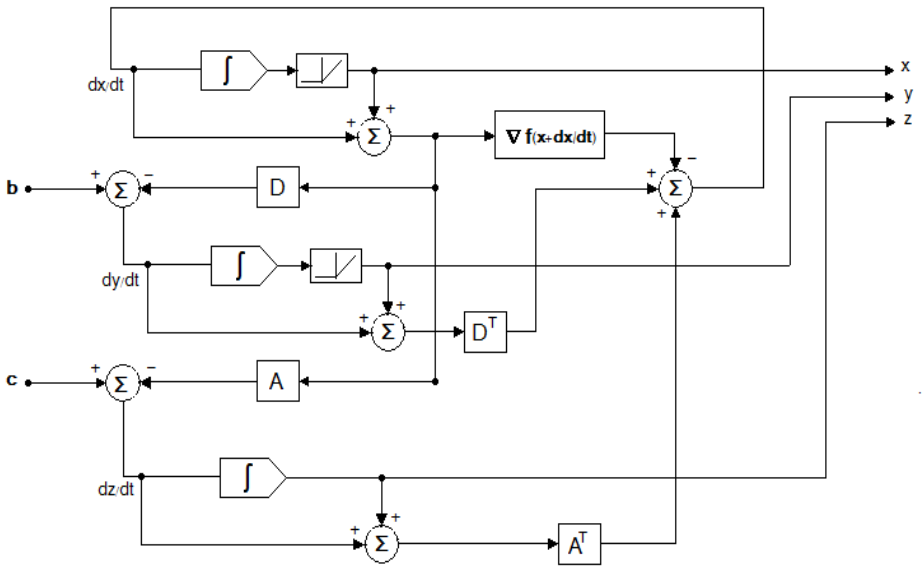

Fig. 1: A simplified block diagram of model $(4)-(6)$.

Theorem 1:

If the solution of the neural network described by the differential equations $(4)-(6)$ converges to a stable state $x(.),\ y(.),\ z(.)$ then $x(.)$ converges to the optimal solution of the primal problem (1) and the Lagrange multipliers $y(.),\ z(.)$ converge to the optimal solution of the dual of the convex programming problem (1).

Proof:

Let x_i be the *i*th component of x, then Eq. (4) can be written as:

$$\frac{dx_i}{dt}=\left(-\nabla f\left(x+k\frac{dx}{dt}\right)+D^T\left(y+k\frac{dy}{dt}\right)+A^T\left(z+k\frac{dz}{dt}\right)\right)_i \qquad if \quad x_i>0, \tag{7}$$

$$\frac{dx_i}{dt}=\max\left(\left(-\nabla f\left(x+k\frac{dx}{dt}\right)+D^T\left(y+k\frac{dy}{dt}\right)+A^T\left(z+k\frac{dz}{dt}\right)\right)_i,\ 0\right) \quad if \quad x_i=0. \tag{8}$$

Condition (8) is to ensure that x will be bounded from below by 0. Let $x^*,\ y^*,\ z^*$ be the limit of $x(t),\ y(t)$ and $z(t)$, respectively. That is

$$\lim_{t\to\infty}x(t)=x^*,\quad \lim_{t\to\infty}y(t)=y^*,\quad \lim_{t\to\infty}z(t)=z^*. \tag{9}$$

By stability of convergence, we have $\frac{dx^*}{dt}=0,\ \frac{dy^*}{dt}=0,\ \frac{dz^*}{dt}=0.$

From Eqs. (7) and (8) we can write

$$0=\left(-\nabla f\left(x^*\right)+D^Ty^*+A^Tz^*\right)_i \qquad if\ x_i^*>0, \tag{10}$$

$$0=\max\left(\left(-\nabla f\left(x^*\right)+D^Ty^*+A^Tz^*\right)_i,\ 0\right) \quad if\ x_i^*=0. \tag{11}$$

In other words:

$$\left(-\nabla f\left(x^*\right)+D^Ty^*+A^Tz^*\right)_i\le 0, \tag{12}$$

$$x_i^*\left(-\nabla f\left(x^*\right)+D^Ty^*+A^Tz^*\right)_i=0, \tag{13}$$

or

$$\left(\nabla f\left(x^*\right)-D^Ty^*-A^Tz^*\right)\ge 0, \tag{14}$$

$$x^{*T}\left(-\nabla f\left(x^*\right)+D^Ty^*+A^Tz^*\right)=0 \tag{15}$$

Similarity, from Eqs. (5) and (6), we have:

$$Dx^* - b = 0, \tag{16}$$

$$Ax^* - c \geq 0, \tag{17}$$

$$z^{*T}\left(Ax^* - c\right) = 0. \tag{18}$$

By KKT conditions in (3) and conditions provided in $(14)-(18)$ we show that x^* and $\left(y^*, z^*\right)$ are the optimal solutions for the problem (1) and its dual problem, respectively. This completes the proof.

The Euler method is used to solve the system of ordinary differential equations given by Eqs. $(4)-(6)$. The following Matlab code describes the discrete implementation of the recurrent neural network model proposed with (4), (5) and (6). Here, the coefficient k is set to be equal to the time step dt for the simplicity of the calculations.

```
for  i = 1:n
dy = dt*(b - D*(x + dx));
y = y + dy;
dz = dt*(- A*(x + dx) + c);
dz = max(z + dz , 0) - z;
z = z + dz;
dx = dt*(- ∇f(x + dx) + D^T *(y + dy) + A^T *(z + dz));
dx = max(x + dx , 0) - x;
x = x + dx;
end;
```

4. Numerical Examples:

To demonstrate the behavior and properties of the proposed recurrent neural network model, three examples are simulated by using the Euler method for time step $dt = 0.1$ and *n*=1000.

In Table 1, 2 and 3 we give various set of initial input values to the novel recurrent neural network model and the corresponding optimal solutions.

In Figs 2-13 trajectories for the different initial vector points are drawn to show the behavior of the convergence of the solutions as the number of iterations is increased.

4.1. Example 1:

Consider the following convex nonlinear programming problem:

$$\min\ f(x) = 0.4x_1 + x_1^2 + x_2^2 - x_1x_2 + 0.5x_3^2 + 0.5x_4^2 + \frac{x_1^3}{30}$$

$$\begin{aligned} s.t. \quad & -0.5x_1 - x_2 + x_4 \geq -0.5, \\ & x_1 + 0.5x_2 - x_3 = 0.4, \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

We have tasted the neural network model proposed in Eqs. (4)-(6) using all four possible initial points (Four possible combination for feasible and infeasible points) for the primal and dual problems. The numerical results for this example are obtained in the Table 1 and Figs. 2-5. It is shown that after 80 iterations the input initial value to the novel neural network model will converge to the stable optimal solutions $x^* = (0.25667228100118\ ,\ 0.28665543799763\ ,\ 0\ ,\ 0)$ and $(y^*, z^*) = (y^* = 0.63327718998817\ ,\ z^* = 0)$ for primal and its dual problem respectively.

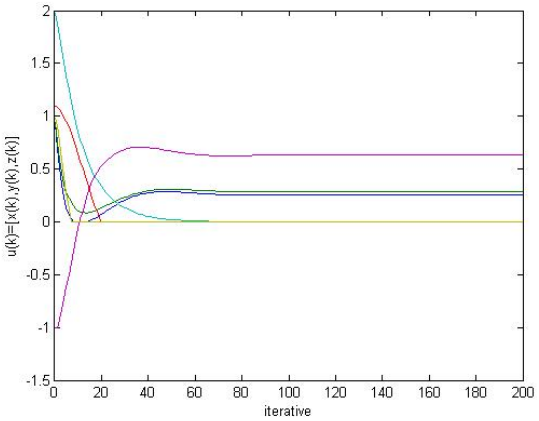

Fig. 2: Trajectories of example 1 problem for the feasible initial vectors $x_0 = (1,1,1.1,2)$ and $(y_0, z_0) = (-1,1)$ using model $(4)-(6)$.

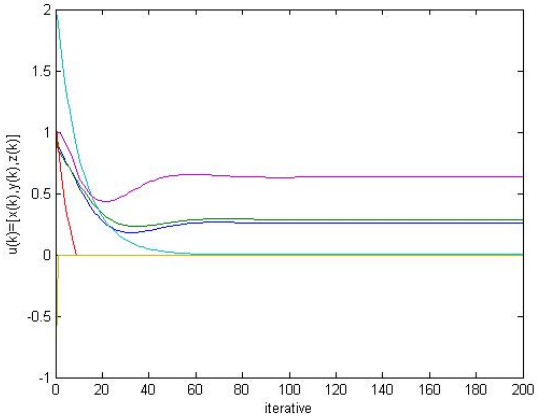

Fig. 3: Trajectories of example 1 problem for the feasible and infeasible initial vectors $x_0 = (1,1,1.1,2)$ and $(y_0, z_0) = (1,-1)$, respectively using model $(4)-(6)$.

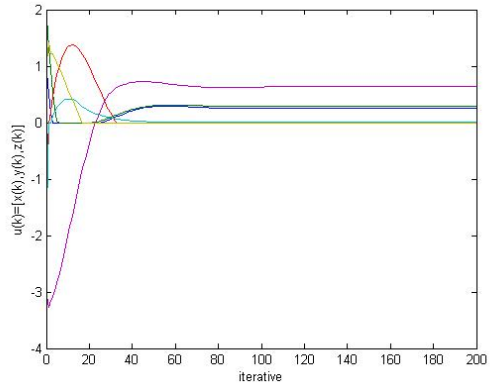

Fig. 4: Trajectories of example 1 problem for the infeasible and feasible initial vectors $x_0 = (1,2,-1,-2)$ and $(y_0, z_0) = (-3,1)$, respectively using model $(4)-(6)$.

Table 1: Numerical results for the primal and dual problems of example 1 using model (4) –(6) for four different initial points (feasible and infeasible).

Initial points			Feasible or infeasible		Optimal solution			Figure
Primal	Dual		Primal	Dual	Primal	Dual		
x_0	y_0	z_0			x^*	y^*	z^*	
1	-1	1			0.25667228100118	0.63327718998817	0	
1			Feasible	Feasible	0.28665543799763			Fig. 2
1.1					0			
2					0.00000000000000			
1	1	-1			0.25667228100118	0.63327718998817	0	
1			Feasible	Infeasible	0.28665543799763			Fig. 3
1.1					0			
2					0.00000000000000			
1	-3	1			0.25667228100118	0.63327718998817	0	
2			Infeasible	Feasible	0.28665543799763			Fig. 4
-1					0			
-2					0.00000000000000			
-1	1	-1			0.25667228100118	0.63327718998817	0	
2			Infeasible	Infeasible	0.28665543799763			Fig. 5
4					0			
3					0.00000000000000			

Table 2: Numerical results for the primal and dual problems of example 2 using model (4) –(6) for four different initial points (feasible and infeasible).

Initial points		Feasible or infeasible		Optimal solution		Figure
Primal	Dual	Primal	Dual	Primal	Dual	
x_0	y_0			x^*	y^*	
0.5	-2			0.99999999999564	-0.00000000001717	
0.5	-1	Feasible	Feasible	0	-0.00000000185370	Fig. 6
0.3				0		
0.7				0.99999999962878		
0.5	2			1.00000000000358	0.00000000001375	
0.5	1	Feasible	Infeasible	0	0.00000000174880	Fig. 7
0.3				0		
0.7				1.00000000035736		
0.6	-3			0.99999999999171	-0.00000000003181	
-0.2	-1	Infeasible	Feasible	0	-0.00000000204367	Fig. 8
0.1				0		
1.5				0.99999999958238		
0.6	2			1.00000000000356	0.00000000001365	
-0.2	1	Infeasible	Infeasible	0	0.00000000147742	Fig. 9
0.1				0		
1.5				1.00000000030191		

Table 3: Numerical results for the primal and dual problems of example 3 using model (4) –(6) for four different initial points (feasible and infeasible).

Initial points		Feasible or infeasible		Optimal solution		Figure
Primal	Dual	Primal	Dual	Primal	Dual	
x_0	y_0			x^*	y^*	
0.3	-3			0.36722148927283	-0.53111404290867	
0.7		Feasible	Feasible	0		Fig. 10
0				0.63277851072717		
0.3	2			0.36722148927283	-0.53111404290867	
0.7		Feasible	Infeasible	0		Fig. 11
0				0.63277851072717		
1	-3			0.36722148927283	-0.53111404290867	
0		Infeasible	Feasible	0		Fig. 12
-1				0.63277851072717		
1	2			0.36722148927283	-0.53111404290867	
0		Infeasible	Infeasible	0		Fig. 13
-1				0.63277851072717		

4.2. Example 2:

Consider the following convex nonlinear programming problem:

$$\min\ f(x)=\frac{3}{2}\left(x_1^2+x_2^2\right)+2\left(x_3^2+x_4^2\right)-Ln\left(x_1x_4\right)+3x_1x_2+4x_3x_4-2x_1-3x_4$$

$$s.t. \quad x_1+x_2=1,$$

$$x_3+x_4=1,$$

$$x_1, x_2, x_3, x_4 \geq 0.$$

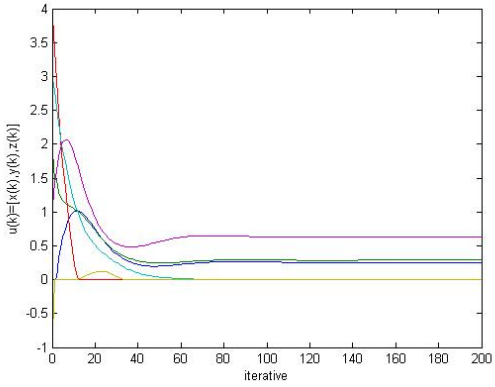

Fig. 5: Trajectories of example 1 problem for the infeasible initial vectors $x_0=(-1,2,4,3)$ and $(y_0,z_0)=(1,-1)$ using model $(4)-(6)$.

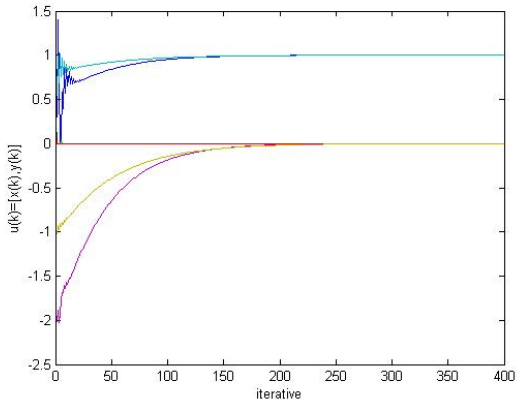

Fig. 6: Trajectories of example 2 problem for the feasible initial vectors $x_0=(0.5,0.5,0.3,0.7)$ and $y_0=(-2,-1)$ using model $(4)-(6)$.

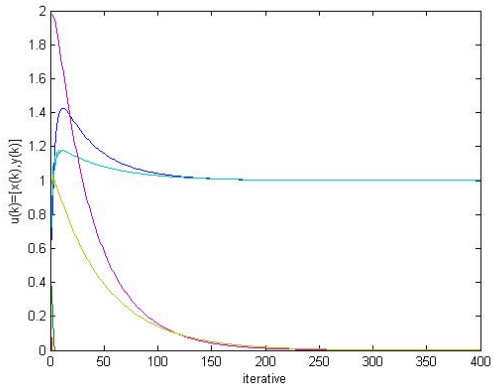

Fig. 7: Trajectories of example 2 problem for the feasible and infeasible initial vectors $x_0=(0.5,0.5,0.3,0.7)$ and $y_0=(2,1)$, respectively using model $(4)-(6)$.

We have tasted the neural network by choosing four arbitrary possible initial points. i.e., all possible combinations for feasible and infeasible starting values for primal and dual problems are considered. The numerical results are gathered in Table 2 and Figs. 6-9 show the behavior of the convergence for the initial point to the optimal solution when the number of iterations is increased. As it is obvious in these figures at most 250 iterations are needed that successfully converge to the correct optimal solution $x^* = (1\ ,\ 0\ ,\ 0\ ,\ 1)$ and $y^* = (0\ ,\ 0)$ for primal and dual problems respectively.

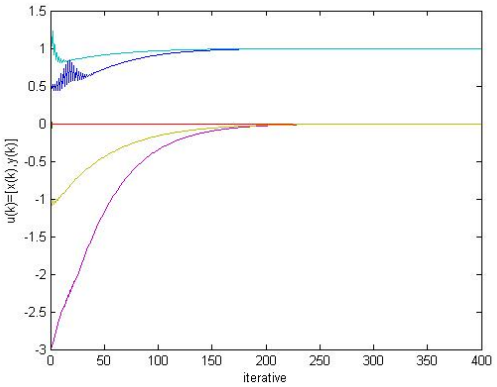

Fig. 8: Trajectories of example 2 problem for the infeasible and feasible initial vectors $x_0 = (0.6, -0.2, 0.1, 1.5)$ and $y_0 = (-3, -1)$, respectively using model $(4)-(6)$.

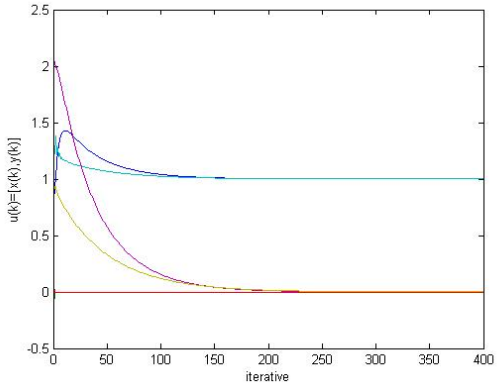

Fig. 9: Trajectories of example 2 problem for the infeasible initial vectors $x_0 = (0.6, -0.2, 0.1, 1.5)$ and $y_0 = (2, 1)$, respectively using model $(4)-(6)$.

4.3. Example 3:

Consider the following convex nonlinear programming problem:

$$\begin{aligned} \min\ & f(x) = (1 - x_1)^2 + (x_1 - x_2)^2 + e^{x_2 - x_3} \\ s.t.\quad & x_1 + x_2 + x_3 = 1, \\ & x_1\ ,\ x_2\ ,\ x_3 \geq 0. \end{aligned}$$

Its optimal solutions are $x^* = (0.36722148927283\ ,\ 0\ ,\ 0.63277851072717)$ and $y^* = -0.53111404290867$ for the primal and its dual problem respectively. In Table 3 we give four different set of initial input values to the neural network model and their related optimal set values. In Figs. 10-13 it is shown that for all four possible initial points (feasible and infeasible) one need not more than 120 iterations to successfully converge to the correct optimal solutions. Simulations for the convergence of the trajectories are shown in Figs. 10-13 for various set of the initial points of Table 3.

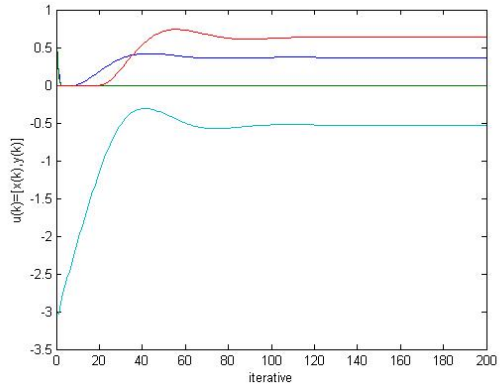

Fig. 10: Trajectories of example 3 problem for the feasible initial vectors $x_0 = (0.3, 0.7, 0)$ and $y_0 = -3$, respectively using model $(4)-(6)$.

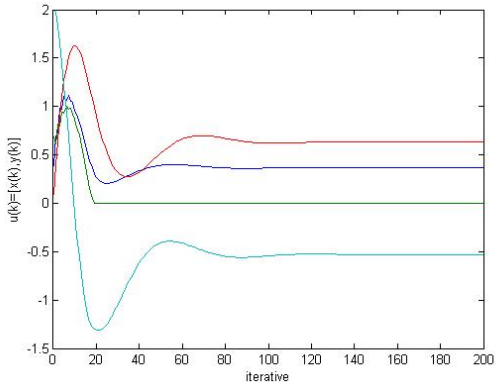

Fig. 11: Trajectories of example 3 problem for the feasible and infeasible initial vectors $x_0 = (0.3, 0.7, 0)$ and $y_0 = 2$, respectively using model $(4)-(6)$.

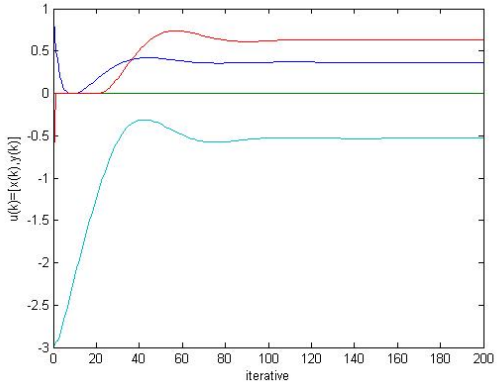

Fig. 12: Trajectories of example 3 problem for the infeasible and feasible initial vectors $x_0 = (1, 0, -1)$ and $y_0 = -3$, respectively using model $(4)-(6)$.

5. Conclusions:

The neural network proposed in this paper has many advantages compared to existing neural networks for solving convex nonlinear programming problems. It converges to the exact solutions of primal and dual problem without requiring any parameter tuning. The new model is simple to use. Another advantage of the new model is that it is very intuitive and can be explained in common sense without formal mathematics.

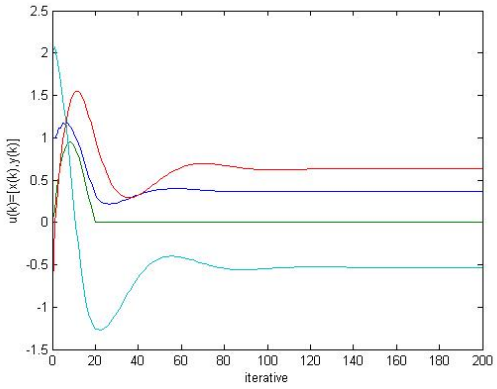

Fig. 13: Trajectories of example 3 problem for the infeasible initial vectors $x_0 = (1,0,-1)$ and $y_0 = 2$, respectively using model $(4)-(6)$.

REFERENCES

Bertsekas, D.P. and J.N. Tsitsiklis, 1989. Parallel and Distributed Computation: Numerical Methods. Prentice-Hall, Englewood Cliffs, NJ.

Cichocki, A., R. Unbehauen, 1993. Neural Networks for Optimization and Signal Processing. Wiley, New York.

Karpinskaya, N.N., 1967. Method of penalty functions and the foundations of Pyne's method. Automation and Remote Contr., 28(1): 124-129.

Kennedy, M.P., L.O. Chua, 1988. Neural networks for nonlinear programming. IEEE Trans. Circuits Syst., 35(5): 554-562.

Li, H., B.S. Manjunath and S.K. Mitra, 1995. Multisensor image fusion using the wavelet transform. Graph. Models Image Process, 57(3): 235-245.

Maa, C.Y., M.A. Shanblatt, 1992. Linear and quadratic programming neural network analysis. IEEE Trans. Neural Netw, 3(4): 580-594.

Malek, A. and A. Yari, 2005. Primal-dual solution for the linear programming problems using neural networks. Appl. Math. Comput., 167: 198-211.

Malek, A. and H.G. Oskoei, 2005. Numerical solutions for constrained quadratic problems using high-performance neural networks. Appl. Math. Comput., 169: 451-471.

Malek, A. and M. Alipour, 2007. Numerical solution for linear and quadratic programming problems using a recurrent neural network. Appl. Math. Comput., 192: 27-39.

Oskoei, H.G., A. Malek and A. Ahmadi, 2007. Novel artificial neural network with simulation aspects for solving linear and quadratic programming problems. Appl. Math. Comput., 53: 1439-1454.

Pyne, I.B., 1956. Linear programming on an electronic analogue computer. Trans. Am. Ins. Elect. Eng., 75: 139-143.

Rybashov, M.V., 1965. The gradient method of solving convex programming problems on electronic analog computers. Automation and Remote Contr., 26(11): 1886-1898.

Rybashov, M.V., 1965. Gradient method of solving linear and quadratic programming problems on electronic analog computers. Automation and Remote Contr., 26(12): 2079-2089.

Xia, Y.S., 1996. A new neural network for solving linear and quadratic programming problems. IEEE Trans. Neural Netw., 7(6): 1544-1547.

Yashtini, M. and A. Malek, 2007. Solving complementarity and variational inequalities problems using neural networks. Appl. Math. Comput., 190(1): 216-230.

Yashtini, M. and A. Malek, 2008. A discrete-time neural network for solving nonlinear convex problems witrh hybrid constraints. Appl. Math. Comput., 195: 576-584.

Zak, S.H., V. Upatising and S. Hui, 1995. Solving linear programming problems with neural networks: Acomparative study. IEEE Trans. Neural Netw, 6(1): 94-104.