International Journal of
Mathematical Modelling & Computations
Vol. 12, No. 01, Winter 2022, 15- 25

A Method for Solving Nonsmooth Pseudoconvex Optimization

M. Bala Seyed Ghasir[a,*], A. Heydari[a] and M. A. Badamchizadeh[b]

[a]*Department of Mathematics, Payame Noor University (PNU), P.O. Box 19395-4697, Tehran, Iran,*
[b]*Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran.*

Abstract. In this paper, a two layer recurrent neural network(RNN) is shown for solving nonsmooth pseudoconvex optimization . First it is proved that the equilibrium point of the proposed neural network(NN) is equivalent to the optimal solution of the orginal optimization problem. Then, it is proved that the state of the proposed neural network is stable in the sense of Lyapunov, and convergent to an exact optimal solution of the original optimization. Finally two examples are given to illustrate the effectiveness of the proposed neural network.

Received: 24 August 2021, Revised: 10 October 2021, Accepted: 20 January 2022.

Keywords: Recurrent neural network; Nonsmooth pseudoconvex; Optimization, Global convergence.

AMS Subject Classification: 90BXX, 90CXX.

Index to information contained in this paper

1. Introduction

Non linear programming problems(NLPP) is applicable in scientific and engineering. RNN is used for solving optimization problem in past years. The first RNN method for solving linear programming problems has been used by Tank and Hopfield in [14], which is turning point for extension of neural networks . A novel RNN for nonlinear convex programming presented by Xia and Wang [15] .While most researchers like to study on convex problems, they concluded that nonconvex

*Corresponding author. Email:phd.seyedghasir.m@pnum.ac.ir programming is more applicable than convex programming. Pseudoconvex programming are more significiant than other nonconvex programming. Hu and Wang [5] used (NN) to solve pseudo monotone problems. Solving a one-layer RNN pseudoconvex optimization has been proposed by Liu et al. [9] . A two-layer RNN for solving nonsmooth convex optimization problems has been proposed by Qin and Xue [13]. Qingfa Li and his coworkers [7], proposed neural network(NN) for solving nonsmooth pseudo convex programming. Qin and his coworkers [12], used one-layer RNN for solving nonsmooth pseudoconvex programming with linear constraints. Guocheng Li et al. [8], proposed one-layer NN for solving nonconvex programming, Their method based on penalty function. Xue and Bian [16], proposed a RNN based on gradient method. Cheng et al. [3], proposed a (RNN) for solving convex problems. Bian and his coworkers [1], proposed RNN for solving pseudoconvex optimization problems with nonsmooth constraints.In this paper, it is used a new (RRN) method for solving pseudoconvex optimization problems with general constraints. Differential inclusion are used to solve nonsmooth pseudoconvex programming subject to general constraints. Our method has two layer structure. Compared with other exisiting neural network methods, The Rate of convergence is fast in this proposed neural network, Because it is not used any penalty function in it's structure. This method is applicable for vast various of pseudoconvex optimization problems.

In this section,some definition and lemmas are considered,which is applicable in this paper.

Definition 1.1 ([11]) Suppose a given point $x_0 \in \mathbb{R}^n$, and f be Lipschitz near this point and $\Game$ is other vector in $\mathbb{R}^n$. Directional derivative of f at x_0 in the direction $\Game$, denoted $\dot{f}(x_0; \Game)$, is defined follows:

$$\dot{f}(x_0; \Game) = \lim_{\substack{y \to x_0 \\ t \to 0}} \sup \frac{f(y + t\Game) - f(y)}{t}$$

The clarke subdifferential of f at x_0 is given by

$$\partial f(x_0) = \{\xi \in \mathbb{R}^n \quad : \quad \dot{f}(x_0; \Game) \geqslant \langle \xi, \Game \rangle, \quad \forall\ \Game \in \mathbb{R}^n\}$$

Definition 1.2 ([11]) f is regular at x when for all $\Game \in R^n$, the usual one-sided directional derivative $f^{'}(x; \Game)$ exist and $f^{'}(x; \Game) = \lim_{t \to 0^+} \sup \dfrac{f(y + t\Game) - f(y)}{t}$
So $f^{'}(x; \Game) = \dot{f}(x; \Game)$.

Regular-function has a very well known property, which has been used in many papers.

Lemma 1.3 *([13]). If $D : \mathbb{R}^n \to \mathbb{R}$ is regular and $x(t) : \mathbb{R} \to \mathbb{R}^n$ is continuous, then, $x(t)$ and $D\big(x(t)\big) : \mathbb{R}^n \to \mathbb{R}$ are differentiable,which is defined as follows:*

$$\dot{D}\big(x(t)\big) =< \psi, \dot{x}(t) > \quad \forall\ \psi \in \partial D\big(x(t)\big) \qquad \textit{for } a \cdot e \cdot\ t \in [0, +\infty).$$

2. RNN design

Definition 2.1 ([11]). Suppose $X \subseteq \mathbb{R}^n$ is a nonempty convex set, for any distinct points $a, b \in X$, pseudoconvex function $f : X \to \mathbb{R}$ on X is defined if

$$\exists\ u \in \partial f(x) \quad : \quad u^T(b - a) \geqslant 0 \Rightarrow f(b) \geqslant f(a).$$

Consider the following nonlinear optimization problem:

$$\begin{aligned} \min \quad & f(x) \\ \textit{subject to} \quad & g_i(x) \leqslant 0 \\ & Ax = b \end{aligned} \tag{1}$$

where x is the optimization variable, f is the objective function, $g_i(i = 1, 2, ..., m)$ are the inequality constraint functions.

Suppose that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ and the constraint functions $g : \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable at a point x^*. If x^* is a local optimum and the optimization problem satisfies some regularity conditions, then there exist constans $\kappa_i(i = 1, ..., m)$ and $\lambda_j(j = 1, ..., l)$ called K.K.T multipliers, primal feasibility is defined as follows min $f(x)$:

$$\begin{aligned} & -\bigtriangledown f(x^*) = \sum_{i=1}^{m} \kappa_i \bigtriangledown g_i(x^*) + \sum_{j=1}^{l} \lambda_j \bigtriangledown (Ax - b) \\ & g_i(x^*) \leqslant 0, for \qquad i = 1, 2, ..., m \\ & Ax - b = 0 \end{aligned} \tag{2}$$

For writing Dual feasibility, we define $\kappa_i \geqslant 0$, for $i = 1, 2, ..., m$.
With an extra multiplier $\kappa_0 \geqslant 0$, which may be zero as long as $(\kappa_0, \kappa, \lambda) \neq 0$, in front of $\bigtriangledown f(x^*)$, the K.K.T stationary conditions turn in to

$$\begin{aligned} & \kappa_0 \bigtriangledown f(x^*) + \sum_{i=1}^{m} \kappa_i \bigtriangledown g_i(x^*) + \sum_{j=1}^{l} \lambda_j \bigtriangledown (Ax - b) \\ & \kappa_i . g_i(x^*) = 0, i = 1, 2, ..., m \end{aligned} \tag{3}$$

Which are called the fritz john conditions,This optimality conditions holds without consraint qualifications and it is equivalent to the optimality condition K.K.T. The K.K.T conditions belong to a wider class of the first-order necessary conditions which allow for nonsmooth functions using. According to the K.K.T conditions, x^* is a solution of (1) if and only if there exist λ^* and κ^* such that $(x^*, \lambda^*, \kappa^*)$ satisfies the following conditions:

$$\begin{aligned} (\bigtriangledown f(x) + \bigtriangledown g(x)\bar{\kappa}) + A^T\lambda - \kappa &= 0 \\ Ax &= b \end{aligned} \tag{4}$$

$$x \geqslant 0, \kappa \geqslant 0, x^T\kappa = 0 \tag{5}$$

from (4), we have $x = x - \bigtriangledown f(x) - \bigtriangledown g(x)\bar{\kappa} - A^T\lambda + \kappa$, That is

$$Ax = Ax - A \bigtriangledown f(x) - A \bigtriangledown g(x)\bar{\kappa} - AA^T\lambda + A\kappa = b.$$

Thus

$$A^T\lambda = A^T(AA^T)^{-1}(Ax - b) - A^T(AA^T)^{-1}A(\bigtriangledown f(x) + \bar{\kappa} \bigtriangledown g(x) - \kappa) - \kappa \tag{6}$$

Substituting (6) in (4), we have:

$$(\bigtriangledown f(x) + \bigtriangledown g(x)\bar{\kappa} - \kappa) + A^T(AA^T)^{-1}(Ax - b) = 0 \tag{7}$$

Let $Q = A^T(AA^T)^{-1}A$ and $\alpha = A^T(AA^T)^{-1}$, then the above equation can be written as:

$$(I - Q)(\bigtriangledown f(x) + \bigtriangledown g(x)\bar{\kappa} - \kappa) + \alpha(Ax - b) = 0 \tag{8}$$

In addition, by the projection theorem,(5) is equivalent to solving the following equations:

$$(\kappa.g(x) - x)^+ - \kappa.g(x) = 0 \tag{9}$$

So, solving the problem (4) is equivalent to solving the following equations:

$$\begin{cases} (I - Q)(\bigtriangledown f(x) + \bar{\kappa} \bigtriangledown g(x)) + \alpha(Ax - b) = 0 \\ (\bar{\kappa} - x)^+ - \bar{\kappa}^+ = 0 \end{cases} \tag{10}$$

Where $(\kappa)^+ = ([\kappa_1]^+, [\kappa_2]^+, ..., [\kappa_n]^+)^T$,$(\kappa_i)^+ = \max\{\kappa_i, 0\}$.
Based on (10), we proposed a new neural network for solving (5) as follows:

$$\begin{cases} (I - Q)(\bigtriangledown f(x) + \bar{\kappa} \bigtriangledown g(x)) + \alpha(Ax - b) = 0 \\ \dot{\kappa}(t) = -\kappa(t) + (\kappa(t).g(x(t)))^+ \end{cases} \tag{11}$$

From above analysis, it is easy to see that if (x^*, κ^*) is an equilibrium of system (11), then x^* is an optimal solution of (5). So, the proposed two-layer RNN for solving nonsmooth optimization has the following structure:

$$\begin{cases} \dot{x}(t) \in -(I - Q)[\partial f\big(x(t)\big) + \partial g\big(x(t)\big)^T \bar{\kappa}] - A^T(Ax(t) - b) \\ \dot{\kappa}(t) = -\kappa(t) + \bar{\kappa}(t) \end{cases} \tag{12}$$

where $\bar{\kappa(t)} = \big(\kappa(t).(g(x(t))\big)$, I is the identity matrix, $Q = A^T \times (AA^T)^{-1} \times A$, $\partial g\big(x(t)\big)^T = \big(\partial g_1(x(t))\big), \partial g_2\big(x(t)\big), \ldots, \partial g_p\big(x(t)\big)\big)^T$ and $(t)^+ = \max\{t, 0\}$.
For any initial point $\kappa(0) = \kappa_0, x(0) = x_0, t \in [0, T)$. There exist at least a state $\big(x(t), \kappa(t)\big)^T$ of neural network

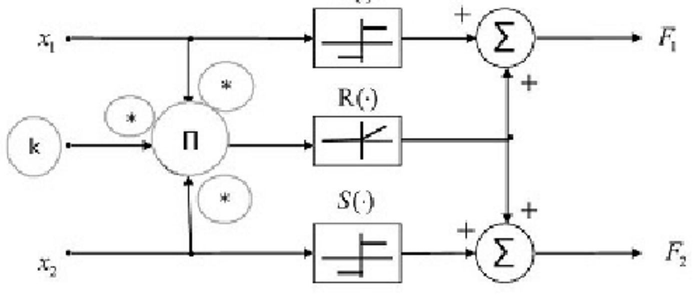

Figure 1. Block diagram of F by circuits [13].

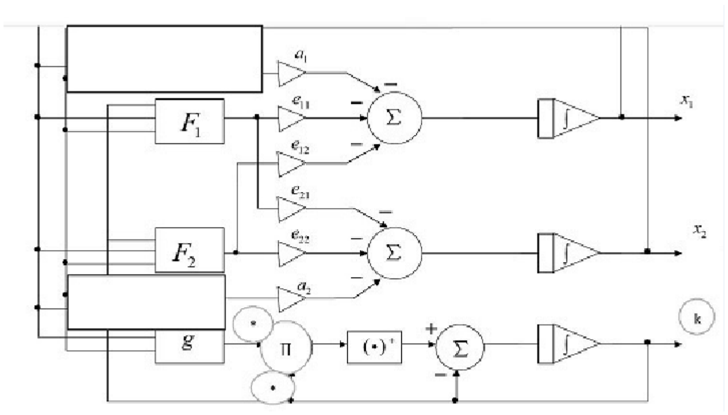

Figure 2. Block diagram of (12) for optimization problem (1) [13].

3. Convergency

In this section convergency of proposed RNN is analysed. it is shown that the state of RNN is exponentially convergent and stability in the sense of Lyapunov is discussed.
Assumption 1. Let $f : \mathbb{R}^n \to \mathbb{R}$ is regular and pseudoconvex and $A \in \mathbb{R}^{m\times n}$ is full-rank and $rank(A) = m \leqslant n$.

Theorem 3.1 *If Assumption 1 is being established, The proposed (RNN) to the feasible region $X = \{x \in \mathbb{R}^n \mid Ax = b, g(x) \leqslant 0\}$ is exponentially convergent.*

Proof Since $\partial f(x)$, $\partial g(x)$ are upper semicontinious with non empty convex compact values, then for any initial point $x_0 \in \mathbb{R}^n$. Let $R(x) = \frac{1}{2}||Ax - b||_2^2$.
Obviously, $R(x)$ is convex, differentiable, and $\nabla R(x) = A^T(Ax - b)$. According to the chain rule and by (20), we have:

$$\frac{d}{dt}R(x(t)) = \nabla R(x(t))^T \cdot \dot{x}(t) = (Ax - b)^T \times A\dot{x}(t) \tag{13}$$

$$\frac{d}{dt}R(x(t)) = (Ax - b)^T \times A[-(I - Q)\big(\partial f(x) + \partial g(x)^T \times \bar{\kappa}\big) - A^T(Ax(t) - b)],$$

then there exist $u(t) \in \partial f(x(t))$, $q(t) \in \partial g(x(t))$, where

$$\frac{d}{dt}R(x(t)) \leqslant \sup(Ax - b)^T A[-(I - Q)(u(t) + q^T\bar{\kappa}) - A^T(Ax - b)].$$

Since,

$$\begin{aligned} A(I - Q) &= A(I - A^T(AA^T)^{-1}A) = A - A = 0 \\ &= -(Ax - b)^T AA^T(Ax - b) \leqslant -\lambda_m(AA^T)||Ax - b||^2 \\ &= -2\lambda_m(AA^T)R(x(t)), \end{aligned} \tag{14}$$

where $\lambda_m(AA^T)$ is the maximum eigenvalue of AA^T. It is obvious that $\lambda_m(AA^T) > 0$, since A is a full row-rank matrix. Then,

$$R(x(t)) \leqslant e^{-2\lambda_m(AA^T)}R(x_0) \tag{15}$$

If $x_0 \in X$, then $R(x_0) = 0$. So $R(x(t)) = 0$ for $t \geqslant 0$. ■

Theorem 3.2 *If Assumption 1 is hold, then for any initial point $x_0 \in X$, It is proved that the state of the proposed (NN) is stable in the sense of Lyapunov, and convergent to an exact optimal solution of the original optimization.*

Proof Let $(x^*, \kappa^*)^T$ be an equilibrium point of proposed neural network, that is, there exist $u^* \in \partial f(x^*)$, $q^* \in \partial g(x^*)$ and $\eta^* \in (Ax^* - b)$ such that

$$\begin{cases} 0 = -(I-Q)(u^* + {q^*}^T \kappa^*) + A^T \cdot \eta^* \\ \dot{\kappa}(t) = -\kappa^*(t) + \big(\kappa^* \cdot g\big(x(t)\big)\big)^+, \end{cases} \tag{16}$$

since $(I - Q) = (I - Q)^2$ and

$$\begin{aligned} \big(x(t) - x^*\big)^T \cdot Q &= \big(x(t) - x^*\big)^T \cdot A^T(A^TA)^{-1}A \\ &= \big(Ax(t) - Ax^*\big)^T \cdot (AA^T)^{-1}A = 0 \end{aligned} \tag{17}$$

Construct the following Lyapunov function:

$$\begin{aligned} M(x,\kappa) =& f(x) - f(x^*) + \frac{1}{4}||x - x^*||^4 + \frac{1}{4}||\kappa - \kappa^*||^4 \\ & -(x - x^*)^T(u^* + {q^*}^T)\kappa^*, \end{aligned} \tag{18}$$

for almost all $t \geqslant 0$. Chain rule is used:

$$\begin{aligned} &\frac{d}{dt}\big(M(x(t), \kappa(t))\big) \\ &= \psi^T(\dot{x}(t)) \\ &= [u(t) + q^T\bar{\kappa} - u^* - {q^*}^T\kappa^* + x(t) - x^*]^T\dot{x}(t) - (\bar{\kappa} - \kappa) \\ &= [u(t) + q^T\bar{\kappa} - u^* - {q^*}^T\kappa^* + x(t) - x^*]^T][-(I-Q)[u(t) + q^T\bar{\kappa}] \\ &\qquad\qquad\qquad\qquad\qquad\qquad - A^T(Ax(t) - b)] \\ &= -||(I-Q)[u(t) + q^T\bar{\kappa}]||^2 - (u^* + {q^*}^T\kappa^*)(I-Q)(u(t) + q^T\bar{\kappa}) \\ &\quad - (x(t) - x^*)(I-Q)[u(t) + q^T\bar{\kappa}] \\ &= -\|\dot{x}(t)\|^2 - (x(t) - x^*)(u(t) - u^*) - (x(t) - x^*)(q^T\bar{\kappa} - q^{T^*}\kappa^*) - (\bar{\kappa} - \kappa) \\ &= -\|\dot{x}(t)\|^2 - \|\dot{\kappa}(t)\|^2 - (x(t) - x^*)(u(t) - u^*) - (x(t) - x^*)(q^T\bar{\kappa} - q^{T^*}\kappa^*). \end{aligned} \tag{19}$$

Since $\{(x(t) - x^*)(u(t) - u^*) \geqslant 0; (x(t) - x^*)(q^T\bar{\kappa} - q^{T^*}\kappa) \geqslant 0\}$. Then we have: $\frac{d}{dt}\big(M(x(t), \kappa(t))\big) \leqslant 0$. Since $0 \leqslant M\big(x(t), \kappa(t)\big) \leqslant M\big(x(0), \kappa(0)\big) < +\infty$ for $t > 0$, Otherwise, since M is bounded, then for any initial point $(x_0, \kappa_0)^T \in \mathbb{R}^n \times \mathbb{R}^p$, the state $\big(x(t), \kappa(t)\big)^T$ of neural network (12) is bounded, which shows that $\big(x(t), \kappa(t)\big)^T$ exsit for $t \in [0, +\infty)$.
Next, Convergency to an equilibrium point of neural network (12) is proved.
Suppose

$$R(x, \kappa) = \inf\big\{||(I-Q)(u(t) + q^T\bar{\kappa})||^2 + \frac{1}{4}||\kappa - \kappa^*||^4\big\} \tag{20}$$

It is obvious that $R(x,\kappa)=0$. If (x,κ) is an equilibrium point of neural network (20). Since $\big(x(t),\kappa(t)\big)^T$ is bounded, there exists a convergent subsequence $\{\big(x(t_k),\kappa(t_k)\big)^T \mid 0 \leqslant t_1 \leqslant t_2 \leqslant \ldots\}$, and $t_k \to +\infty$, such that $\big(x(t_k),\kappa(t_k)\big)^T \to (x^*,\kappa^*)^T$. It is clear that $Ax^*=b$.
Next, it is proved that $R(x^*,\kappa^*)=0$, It shows that $(x^*,\kappa^*)^T$ is an equilibrium point of neural network (12).
Because $\big(x(t),\kappa(t)\big)^T$ is bounded, there exsit $L>0$ such that $||\dot{x}(t)||+||\dot{\kappa}(t)|| \leqslant L$, for all $t \geqslant 0$, suppose $R(x^*,\kappa^*) \neq 0$, then

$$R(x^*,\kappa^*) > 0. \tag{21}$$

Similarity to proof ([13])

$$(x,\kappa) \in B(x^*,\kappa^*;\delta) = \{(x,\kappa) \in \mathbb{R}^n \times \mathbb{R}^p \quad : \quad ||x-x^*||+||\kappa-\kappa^*|| < \delta\}, \tag{22}$$

Since $\big(x(t_k),\kappa(t_k)\big)^T \to (x^*,\kappa^*)^T$. There exist $D>0$, such that

$$||x(t_k)-x^*||+||\kappa(t_k)-\kappa^*|| < \frac{\delta}{2}, \quad \forall k > D.$$

For $t \in \big[t_k - \frac{\delta}{4S}, t_k + \frac{\delta}{4S}\big]$ and $k > D$, then

$$\begin{aligned}||x(t)-x^*||+||\kappa(t)-\kappa^*|| \leqslant{}& ||x(t)-x(t_k)||+||x(t_k)-x^*|| \\ &+ ||\kappa(t)-\kappa(t_k)||+||\kappa(t_k)-\kappa^*||.\end{aligned}$$

Thus

$$||x(t)-x^*||+||\kappa(t)-\kappa^*|| \leqslant L|t-t_k| + \frac{\delta}{2} \leqslant \delta. \tag{23}$$

By (20), $R\big(x(t),\kappa(t)\big) > \varepsilon$ for all $t \in \big[t_k - \frac{\delta}{4S}, t_k + \frac{\delta}{4S}\big]$ and $k > D$, So

$$\begin{aligned}\int_0^{+\infty} R\big(x(t),\kappa(t)\big)dt &\geqslant \int_{U_{k\geqslant D}\big[t_k-\frac{\delta}{4S},t_k+\frac{\delta}{4S}\big]} R\big(x(t),\kappa(t)\big)dt \\ &\geqslant \int_{U_{k\geqslant D}\big[t_k-\frac{\delta}{4S},t_k+\frac{\delta}{4S}\big]} \varepsilon dt \\ &= \sum_{U_{k\geqslant D}} \frac{\delta}{2S}\cdot\varepsilon = +\infty.\end{aligned} \tag{24}$$

On the other hand, from (19) , there exists M_0 such that $\lim\limits_{t\to+\infty} M\big(x(t),\kappa(t)\big) =$

M_0. So, by (12), we have

$$\int_0^{+\infty} R\big(x(t),\kappa(t)\big)dt = \lim_{s\to+\infty}\int_0^{s} R\big(x(t),\kappa(t)\big)dt \tag{25}$$
$$\leqslant -\lim_{s\to+\infty}\int_0^{s} \dot{M}\big(x(t),\kappa(t)\big)dt$$
$$= -\lim\big[M\big(x(t),\kappa(t)\big) - M\big(x(0),\kappa(0)\big)\big]$$
$$= M\big(x(0),\kappa(0)\big) - M_0 < +\infty.$$

Clearly, it contradicts with (24). Since $R(x^*,\kappa^*) = 0$, then an equilibrium point of neural network (12) is $(x^*,\kappa^*)^T$, and an optimal solution to the nonlinear pseudoconvex programming (1) is x^* .
There exist $u^* \in \partial f(x^*), q^* \in \partial g(x^*)$ such that

$$\begin{cases} 0 = (I-Q)(u^* + {q^*}^T\kappa^*) \\ 0 = -\kappa^* + (\kappa^* \cdot g(x^*))^+ \\ Ax^* = b \end{cases} \tag{26}$$

■

4. Applications and examples

Two examples are considerd the effectiveness of the proposed neural network.

Example 4.1 ([2]) Consider the following pseudoconvex programming

$$\begin{aligned} &\min\ (x_1 - x_2 + x_3)^2 + |x_1^2 + 2x_3| + |x_2 - 1| + 20(x_3 - 2)^2 + e^{-x_1-x_2-x_3} \\ &s.t.\ \ Ax = b, \\ &\qquad g_1(x_1,x_2,x_3) = -6x_2 - 4x_3 + x_1^2 + 0.5e^{|x_1-1|} + 2 \leqslant 0, \\ &\qquad g_2(x_1,x_2,x_3) = x_3^3 - 2 \leqslant 0, \end{aligned}$$

in which $A = \begin{bmatrix}1\ 3\ 0.2\end{bmatrix}$, $b = \begin{bmatrix}1\end{bmatrix}$. For solving this model with proposed method, firstly Q and $I - Q$ are calculated

$$Q = A^T \times (AA^T)^{-1} \times A = \begin{bmatrix} 0.0996\ 0.2988\ 0.0199 \\ 0.2988\ 0.8964\ 0.0598 \\ 0.0199\ 0.0598\ 0.0040 \end{bmatrix},$$

$$\big[e_{ij}\big]_{3\times3} = I - Q = \begin{bmatrix} 0.9004 & -0.2988 & -0.0199 \\ -0.2988 & 0.1036 & -0.0598 \\ -0.0199 & -0.0598 & 0.9960 \end{bmatrix}.$$

It can be seen in Figure 1 that equilibrium of this example is $(0, 0.24934, 1.2599, -3.6)$, so the optimal solution of this example is $(0, 0.24934, 1.2599)$, $f(x^*) = 15.4671$.
In comparison with differential inclusion-based methods ([10]), it can be concluded that our new proposed method has the following advantage: As it canbe found,

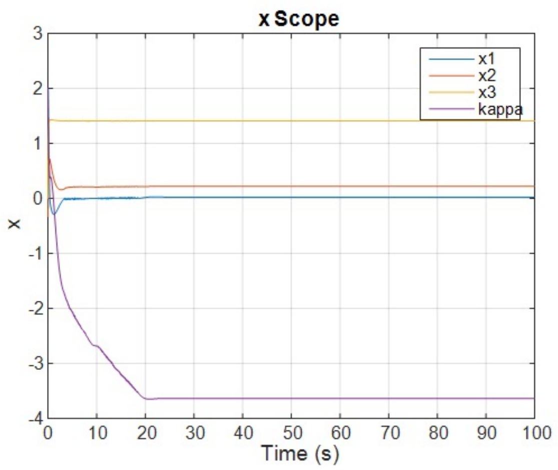

Figure 3. Transient behavior of the states $(x_1(t), x_2(t), x_3(t), \kappa(t))$ of the proposed neural network.

accuracy of the solution and convergence for the penalty method ([10]), depend on selecting the suitable penalty value. In this example, for penalty value $p = 5$ state trajectories converges to an equilibrium point which is not an optimal solution, otherwise, for $p = 15$, state trajectories converges to an equilibrium point whith good accuracy.The new proposed method doesnot need to choose any penalty value for solving problem. In comparison with differential inclusion-based methods ([2, 10]), shows that our proposed method has more accuracy. It can be seen that covergence of state trajectories to the optimal solution in our method is faster than in the penalty-based method ([10]), Because our proposed method has better performance in CPU run time with the there layer neural network ([2]), According to table 2 in ([4]),CPU Run time in ([2]) is 5.4 and in ([10]) is 4.5, But in our method CPU Run time is 1.84.

Example 4.2 Consider the following nonconvex programming

$$\begin{aligned} &\min\ (x_1 + x_2)^2 + cos(x_2) + |x_3 - 2|^3 \\ &s.t.\ \ (x_1 - 4)^2 + x_2 \leqslant 100, \\ &\qquad Ax = b, \end{aligned}$$

in which $A = \begin{bmatrix} 1\ 1\ 1 \end{bmatrix}$, $b = \begin{bmatrix} 1 \end{bmatrix}$. For solving this model, Firstly Q and $I - Q$ are calculated

$$Q = A^T \times (AA^T)^{-1} \times A = \begin{bmatrix} 0.3333 & 0.3333 & 0.3333 \\ 0.3333 & 0.3333 & 0.3333 \\ 0.3333 & 0.3333 & 0.3333 \end{bmatrix},$$

$$\begin{bmatrix} e_{ij} \end{bmatrix}_{3\times3} = I - Q = \begin{bmatrix} 0.6667 & -0.3333 & 0.3333 \\ -0.3333 & 0.6667 & -0.3333 \\ -0.3333 & -0.3333 & 0.6667 \end{bmatrix}.$$

It can be seen in Figure 2 that equilibrium of this example is $(-4.9, 4.3, 1.6, -1)$, so the optimal solution of this example is $(-4.9, 4.3, 1.6)$, $f(x^*) = 0.0232$.

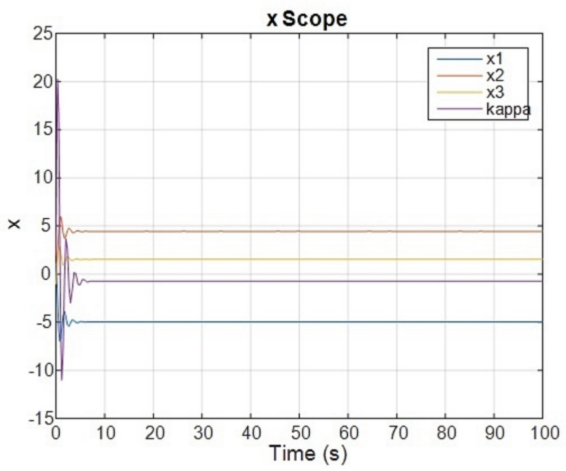

Figure 4. Transient behavior of the states $(x_1(t), x_2(t), x_3(t), \kappa(t))$ of the proposed neural network.

5. Conclusion

In this paper, a two layer recurrent neural network(RNN) is shown for solving non-smooth pseudoconvex optimization . First it is proved that the equilibrium point of the proposed neural network(NN) is equivalent to the optimal solution of the orginal optimization problem. Then, it is proved that the state of the proposed neural network is stable in the sense of Lyapunov, and convergent to an exact optimal solution of the original optimization. Finally two examples are given to illustrate the effectiveness of the proposed neural network.In this paper, it is used a new (RRN) method for solving pseudoconvex optimization problems with general constraints. Differential inclusion are used to solve nonsmooth pseudoconvex programming subject to general constraints. Our method has two layer structure. Compared with other exisiting neural network methods, The Rate of convergence is fast in this proposed neural network, Because it is not used any penalty function in it's structure. This method is applicable for vast various of pseudoconvex optimization problems.

References

[1] W. Bian, L. Ma, S. Qin and X. Xue, Neural network for nonsmooth pseudoconvex optimization with general convex constraints, Neural Netw., **101** (2018) 1–14.

[2] L. Cheng, Z.-G. Hou, Y. Lin, M. Tan, W. C. Zhang and F.-X. Wu, Recurrent neural network for non-smooth convex optimization problems with application to the identification of geneic requlatory networks, IEEE Trans. Neural Netw., **22 (5)** (2011) 714–726.

[3] L. Cheng, Z.-G. Hou and M. Tan, A delayed projection neural network for solving linear variational inequalities, IEEE Trans. Neural Netw., **20 (6)** (2009), 915–925.

[4] M. J. Ebadi, A. Hosseini and M. M. Hosseini, A projection type stepest descent neural network for solving a class of nonsmooth optimization problems, Neurocomputing, **235 (26)** (2017) 164–181.

[5] X. Hu and J. Wang, Solving pseudoconvex monotone variational inequalities and pseudoconvex optimization problems using the projection neural network, IEEE Trans. on Neural Netw., **17** (2006) 1487–1499.

[6] X. Ju, C. Li, X. He and G. Feng, An inertial projection neural network for solving inverse variational inequalities, Neurocomputing, **406 (17)** (2020) 99–105.

[7] Q. Li, Y. Liu and L. Zhu, Neural network for nonsmooth pseudoconvex optimization with general constraints, Neurocomputing, **131** (2014) 336–347.

[8] G. Li, Z. Yan and J. Wang, A one-layer recurrent neural network for constrained nonconvex optimization, Neural Netw., **61** (2014) 10–21.

[9] Q. Liu, Z. Guo and J. Wang, A one-layer recurrent neural network for constrained pseudoconvex optimization and its application for dynamic portfolio optimization, Neural Netw., **26** (2012) 99–109.

[10] Q. Liu and J. Wang, A one-layer recurrent neural network for constrained nonsmooth optimization, IEEE Trans. Syst. Man Cybern. Part B Cybern., **41 (5)** (2011) 1323–1333.

[11] S. Qin, W. Bian and X. Xue, A new one-layer recurrent neural network for nonsmooth pseudoconvex optimization, Neurocomputing, **120** (2013) 655–662.

[12] S. Qin, D. Fan, P. Su and Q. Liu, A simplified recurrent neural network for pseudoconvex optimization subject to linear equality constraints, Commun. Nonlinear Sci. Numer. Simul., **19 (4)**(2014) 789–798.

[13] S. Qin and X. Xue, A two-layer recurrent neural network for nonsmooth convex optimization problems, IEEE Trans. Neural Netw. Learn. Syst., **26 (6)** (2015) 1149–1160.

[14] D. Tank and J. Hopfield, Simple neural optimization networks:and a/d neural convex signal decision circuit, IEEE Trans. Circuits Syst., **33** (1986) 533–554.

[15] Y. Xia and J. Wang, A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints, IEEE Trans. Circuits Syst., **51** (2004) 1385–1394.

[16] X. Xue and W. Bian, Subgradient-based neural networks for nonsmooth convex optimization problems, IEEE Trans. Circuits Syst. I Regul. Pap., **55 (8)** (2008) 2378–2391.