Payame Noor University

Control and Optimization in Applied Mathematics (COAM)
Vol. 7, No. 2, Summer-Autumn 2022 (35-52), ©2016 Payame Noor University, Iran

DOI. 10.30473/coam.2022.64268.1206 (Cited this article)

Research Article

A New Optimization Method Based on Dynamic Neural Networks for Solving Non-convex Quadratic Constrained Optimization Problems

Kobra Mohammadsalahi[1], Farzin Modarres Khiyabani[1*], Nima Azarmir Shotorbani[1]

[1]Department of Mathematics, Tabriz Branch, Islamic Azad University, Tabriz, Iran.

Received: June 02, 2022 ; **Accepted:** November 28, 2022 .

Abstract. This paper presents a capable recurrent neural network, the so-called μRNN for solving a class of non-convex quadratic programming problems. Based on the optimality conditions we construct a new recurrent neural network (μRNN), which has a simple structure and its capability is preserved. The proposed neural network model is stable in the sense of Lyapunov and converges to the exact optimal solution of the original problem. In a particular case, the optimality conditions of the problem become necessary and sufficient. Numerical experiments and comparisons with some existing algorithms are presented to illustrate the theoretical results and show the efficiency of the proposed network.

Keywords. Quadratic programming, Recurrent neural network, Non-convex optimization.

MSC. 90C34; 90C40.

* Corresponding author
iauasrb082@gmail.com, salahi763@gmail.com, azarmir$_n$im@yahoo.com
http : //mathco.journals.pnu.ac.ir

1 Introduction

Quadratic programming problems arise in a wide variety of scientific and engineering applications including regression analysis, image and signal processing, parameter estimation, filter design, robot control, etc. See for example [3, 2, 40] and a study of piecewise linear-quadratic programs by Cui et al. [12]. Optimization problems with nonlinear objective functions are usually approximated by second-order (quadratic) systems and solved approximately by standard quadratic programming (QP) techniques [29, 30]. In modeling many scientific problems, quadratic problems are obtained, such as smoothing quadratic regularization methods [6], minimizing condition number [10] and machine learning [19].

In recent years, convex QP has been studied by many researchers and many good results have been obtained. For example, one can see [14, 16, 45, 48, 49] where several methods for solving degenerate QP, convex quadratic bilevel programming, and convex quadratic minimax problems have been proposed. A major difference between convex and non-convex quadratic programming ($NCQP$) problems is that for the former any local minimizer is also a global minimizer whereas the latter may have many local minimizers. An analytic method for $NCQP$ subject to a set of linear constraints is presented in [4, 8]. Jeyakumar et al. [23] establish Lagrange multiplier conditions for global optimality of general non-convex quadratic minimization problems with quadratic constraints. They also obtain necessary global optimality conditions, which are different from the Lagrange multiplier conditions for special classes of QPs (see [42, 43] for more details). Moreover, Huang et al. [21] and Kong et al. [26] have used faster gradient-free proximal stochastic methods to solve the non-convex non-smooth optimization. QPs can also be solved indirectly using unconstrained optimization methods and optimization algorithms [34, 35, 46]. There are several direct methods to solve CQP and $NCQP$, such as first-order methods [9], interior point algorithms [7], accelerated gradient method [20] and combining stochastic adaptive cubic regularization [39].

The neural networks for solving mathematical programming problems were first proposed by Tank and Hopfild [44, 17]. Their work has inspired many researchers to investigate other neural network models for solving programming problems. One of the efficient methods to solve CQP and $NCQP$ is a recurrent neural network (RNN). The main advantage of RNN to optimization is that they can solve optimization problems in running time at orders of magnitude much faster than the most traditional optimization algorithms [5, 50]. These networks have been used in many scientific applications, such as complex-variable programming problems [28], classifiers with low model complexity [38], and non-smooth constrained pseudo-convex optimization [47].

Xue and Bian [48] developed a project neural network for solving degenerate QP problems with general linear constraints. In the theoretical aspects, the proposed Neural Network (NN) is shown to have complete convergence and finite time convergence. Effati and Ranjbar [14] presented a new NN for solving CQP problems. This model has a simple form, furthermore, it has a good convergence rate with a less number calculation operation than the old models. Besides, Nazemi [37] has used a capable NN for solving strictly CQP problems with general linear constraints. Nonetheless, Malek and Hosseinipour-Mahani [31] in their paper demonstrated that the use of the

$RNNs$ to solve $NCQP$ is efficient. In this work, based on a generalized KKT method, a modified RNN model called $M.RNN$ for a class of $NCQP$ problems involving a so-called Z-matrix has been proposed. By the study of the resulting dynamic system, it is shown that under given assumptions, steady states of the dynamic system are stable [1, 25].

There is similar research in the field of nonlinear and non-convex programming via neural networks. Effati et al. [13] presented an efficient projection neural network for solving bilinear programming problems. Also, Eshaghnezhad et al. [15] used a neurodynamic model to solve the nonlinear pseudo-monotone projection equation and its applications. Nonetheless, there are other types of numerical approaches to solving optimization problems via neural networks. Mansoori and Effati [32, 33] applied a parametric NCP-based recurrent neural network model to solve fuzzy non-convex optimization problems. Leung and Wang [27] proposed a neurodynamic method to solve minimax and bi-objective portfolio problems.

In this paper, an RNN network is designed to solve the $NCQP$ problems. We call this network μRNN. μRNN is similar to $M.RNN$ in reference [31]. The authors in [31] could have designed the $M.RNN$ more easily, thus reducing calculations and proofs. Accordingly, we have designed an RNN as μRNN that is simple and highly efficient. μRNN is stable in the sense of Lyapunov and has a high speed of convergence. Thus μRNN not only solves CQP and $NCQP$, but also has the following advantages:

- μRNN has a simple structure, so it is easy to design and use.
- The convergence rate of μRNN is sometimes equal to the convergence rate of $M.RNN$, and somtimes it is better.

In terms of run time, the μRNN network is similar to the $M.RNN$ network. In some problems where it is necessary to choose a small Rung-Kutta numerical method step length, the run time increases in both methods (as in Example 2). But in many convex and non-convex problems, the run time is about a few seconds.

2 Preliminaries

This section provides the necessary mathematical background used to study the proposed method and its usage. We list some necessary notations and introduce some necessary preliminary results in this section.

- $\|.\|$ denotes the l_2-norm on $\mathbb{R}^n$ ($\|x\| = (\sum_{i=1}^{n} x_i^2)^{1/2}$) and e_i denotes the column vector with a 1 in the i-th coordinate and 0's elsewhere.
- The space of all $n \times n$ symmetric matrices is denoted by S^n.
- For $g : \mathbb{R}^n \to \mathbb{R}$, $\nabla g(x) \in \mathbb{R}$ and $\nabla^2 g(x) \in \mathbb{R}^{n\times n}$ stand for gradient and the Hessian of g at x.
- The notation $A \succcurlyeq 0$ ($A \preccurlyeq 0$) shows that the matrix A is positive (negative) semi-definite.

- If there exists a non-zero vector $x \in \mathbb{R}^n$ such that $x^T Ax < 0$ then $A \not\succcurlyeq 0$.

Consider the following smooth non-convex quadratic optimization problem.

$$\begin{cases} Min\ f(X) \\ s.t:\ g_i(X) \leq 0,\ i = 1,2,\ldots,m, \end{cases} \tag{1}$$

where $f, g_i : \mathbb{R}^n \to \mathbb{R}$ are defined by

$$f(X) = \frac{1}{2}X^T A_f X + b_f^T X + c_f,\ g_i(X) = \frac{1}{2}X^T A_{g_i} X + b_{g_i}^T + c_{g_i},$$

and $S_0 = \{X \in \mathbb{R}^n | g_i(X) \leq 0, i = 1,2,\ldots,m\}$ is the feasible set. We suppose that $A_f \not\succcurlyeq 0$, and define H_f, H_{g_i} for $i = 1,2,\ldots,m$ by

$$H_f = \begin{pmatrix} A_f & b_f \\ b_f^T & 2c_f \end{pmatrix},\ H_{g_i} = \begin{pmatrix} A_{g_i} & b_{g_i} \\ b_{g_i}^T & 2c_{g_i} \end{pmatrix}. \tag{2}$$

Definition 1. A matrix $A \in S^n$ is called a Z-matrix if $a_{ij} \leq 0$ for all $i \neq j$. Therefore any diagonal matrix is a Z-matrix.

In this paper, the RNN will be designed based on the following statement.

Proposition 1. (Jeyakumar et al. [23]) For general non-convex quadratic programming problem (1), let $X^* \in S_0$. If there exists $\lambda = (\lambda_1,\ldots,\lambda_m)^T \in \mathbb{R}_+^m - \{0\}$ such that the conditions

$$\begin{cases} (a) & A_f + \sum_{i=1}^m \lambda_i A_{g_i} \succeq \mathbf{O}, \\ (b) & (A_f x^* + b_f) + \sum_{i=1}^m (\lambda_i A_{g_i} x^* + \lambda_i b_{g_i}) = O, \\ (c) & \sum_{i=1}^m \lambda_i g_i(x^*) = 0, \end{cases} \tag{3}$$

hold, then X^* is a global minimizer of (1).

Remark 1. In the problem (1) when $m = 1$ and the strict feasibility condition holds, conditions (3) are necessary and sufficient conditions [24]. Also, for $m > 1$ the condition (a) of (3) is just a sufficient (not necessary) global optimality condition [31].

Theorem 1. (Jeyakumar et al. [22]) For the non-convex quadratic problem (1), suppose that H_f and H_{g_i}, $i = 1,\ldots,m$ are Z-matrices and the Slater condition holds, that is, there exists $X_0 \in \mathbb{R}^n$ such that $g_i(X_0) < 0, i = 1,\ldots,m$. Then a feasible point X^* is a globally optimal solution if and only if the conditions (3) hold.

Lemma 1. If A is a real square matrix, then:

$$X^T AX = X^T A^T X.$$

Proof. We know that any square matrix A can be written as [18]

$$A = \frac{1}{2}(A + A^*) + \frac{1}{2}(A - A^*) \equiv B + C,$$

where $B = \frac{1}{2}(A + A^*)$ is the Hermitian part of A, and $C = \frac{1}{2}(A - A^*)$ is the skew-Hermitian part of A. We have

$$X^TAX = X^TBX + X^TCX,$$

where $X^TCX = 0$. Thus, if the matrix A is real,

$$X^TAX = \frac{1}{2}X^T(A + A^T)X \Rightarrow X^TAX = X^TA^TX.$$

□

Corollary 1. Without loss of generality, in (1) assume that the matrices A_f and A_{g_i}, $i = 1,\ldots,m$ are symmetric. If the matrix A in X^TAX is not symmetric then we can replace it with $\frac{1}{2}(A + A^T)$.

Consider the following differential equation:

$$\dot{X}(t) = f(X(t)), \ X(t_0) = X_0 \in \mathbb{R}^n. \tag{4}$$

The following classical result on the existence and uniqueness of the solution to (4) holds.

Theorem 2. (Uniqueness and Existence) Assume that g is a continuous mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$. Then for arbitrary $t_0 \geq 0$ and $X_0 \in \mathbb{R}^n$ there exists a local solution $X(t)$, $t \in [t_0, \tau)$ to (4) for some $\tau > t_0$. If g is locally Lipschitzian continuous at X_0 then the solution is unique, and if g is Lipschitzian continuous in $\mathbb{R}^n$ then τ can be extended to ∞.

Proof. See [11]. □

3 Recurrent Neural Network

Based on the optimization conditions (3), we design an RNN that converges to the optimal solutions to the problem (1). Consider

$$\begin{cases} \frac{dX}{dt} = -A_fX - b_f - \frac{1}{2}\sum_{i=1}^m \mu_i^2(A_{g_i}X + b_{g_i}), \\ \frac{d\mu}{dt} = diag(\mu_1,\ldots,\mu_m).g(X), \end{cases} \tag{5}$$

where $g(X) = (g_1(X),\cdots,g_m(X))^T$ and $\mu = (\mu_1,\cdots,\mu_m)^T$. Assuming $y = (X^T,\mu^T)^T$ and

$$\nabla g(X) = (\nabla g_1^T(X),\cdots,\nabla g_m^T(X))^T, \ \nabla g_i = \left(\frac{\partial g_i}{\partial x_1}, \frac{\partial g_i}{\partial x_2},\cdots,\frac{\partial g_i}{\partial x_n}\right)^T.$$

We call system (5), μRNN and it is summarized as follows

$$\dot{y} = K\varphi(y), \tag{6}$$

where

$$\varphi(y) = \begin{pmatrix} -\nabla f(X) - \frac{1}{2}\sum_{i=1}^m \mu_i^2\nabla g_i \\ diag(\mu_1,\cdots,\mu_m)g^T(X) \end{pmatrix}, \ y(t_0) = y_0, \tag{7}$$

and K is an adjusted parameter. A sufficiently large K could accelerate the μRNN.

Lemma 2. If $A_{n\times n}$ and $B_{m\times m}$ are negative definite, then the following matrix is negative definite.

$$\mathcal{F} = \left[\begin{array}{c|c} A_{n\times n} & C_{n\times m} \\ \hline -C^T_{m\times n} & B_{m\times m} \end{array}\right]_{(m+n)\times(m+n)}. \tag{8}$$

Proof. For all $X = (x_1, \cdots, x_n, x_{n+1}, \cdots, x_{m+n})^T$ we have:

$$\begin{aligned} X^T\mathcal{F}X =& X^T\left[\begin{array}{c|c} A & C \\ \hline -C^T & B \end{array}\right]X \\ &= (x_1,\ldots,x_n)A\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + (x_{n+1},\ldots,x_{m+n})B\begin{pmatrix} x_{n+1} \\ \vdots \\ x_{m+n} \end{pmatrix} \\ &\underbrace{+ (x_1,\ldots,x_n)C\begin{pmatrix} x_{n+1} \\ \vdots \\ x_{m+n} \end{pmatrix} - (x_{n+1},\ldots,x_{m+n})C^T\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}}_{=0} \leq 0, \end{aligned}$$

so the matrix $\mathcal{F}$ is negative definite. □

For simplicity of our analysis, we let $K = 1$. An indication of how the neural networks (6) and (7) can be implemented on hardware is provided in Figure 1.

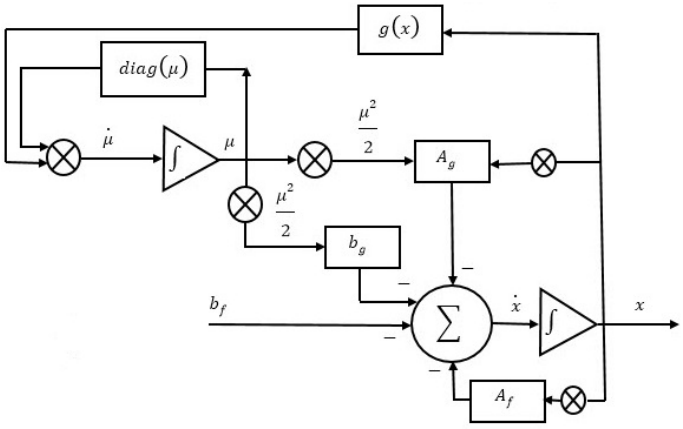

Figure 1: A simplified block diagram for the neural networks (6) and (7).

Theorem 3. Let $\Omega^* \subset \mathbb{R}^{n+m}$ be the set of equilibrium points of the system (6). If $y^* = (X^{*T}, \mu^{*T})^T \in \Omega^*$ where $\mu^* = (\mu_1^*, \cdots, \mu_m^*)^T$, and

$$A_f + \sum_{i=1}^{m} \frac{\mu_i^{*2}}{2} A_{g_i} \succcurlyeq 0,$$

then X^* is a global optimal solution of (1). Also, if H_f and $H_{g_i}, i = 1,\ldots,m$ are Z-matrices and x^* is a global optimal solution of (1), then there exists $\mu^* \in \mathbb{R}^+ - \{O\}$ such that $(X^{*T}, \mu^{*T})^T$ is an equilibrium point of the μRNN.

Proof. Since y^* is an equilibrium point of the μRNN,

$$-A_f X^* - b_f - \frac{1}{2}\sum_{i=1}^{m} \mu_i^2 (A_{g_i} X^* + b_{g_i}) = 0, \forall i = 1,\ldots,m; \mu_i g_i(X^*) = 0. \tag{9}$$

Let $\lambda_i^* = \frac{\mu_i^{*2}}{2}$, (9) satisfies assumptions (3), therefore X^* is a global optimal solution of (1). On the other hand, if H_f and H_{g_i} are Z-matrices then by using Theorem 1, conditions (3) are necessary and sufficient for the optimality of x^*. Therefore, if x^* is a global optimal solution of (1), then there exists $\lambda = (\lambda_1,\ldots,\lambda_m) \geq 0$, $\lambda \neq 0$ such that conditions (3) hold. Substituting $\lambda_i = \frac{\mu_i^2}{2}$ into (3), we have

$$\begin{cases} (a1) & A_f + \sum_{i=1}^{m} \frac{\mu_i^2}{2} A_{g_i} \succeq \mathbf{O}, \\ (b1) & (A_f X^* + b_f) + \sum_{i=1}^{m} \frac{\mu_i^2}{2} (A_{g_i} X^* + b_{g_i}) = O, \\ (c1) & \sum_{i=1}^{m} \frac{\mu_i^2}{2} g_i(X^*) = 0. \end{cases} \tag{10}$$

Since for all $X \in \Omega^*$, we have $g_i(X) \leq 0$, $\mu \geq 0$, so there are two cases, one $\mu_i = 0$ and the other $\mu_i \neq 0$. If $\mu_i = 0$ then $\mu_i g_i(X) = 0$, and if $\mu_i \neq 0$, then $\mu_i g_i(X) = 0$ is established again (from condition $(c1)$). Thus $(X^{*T}, \mu^{*T})^T$ is an equilibrium point of the μRNN. □

4 Stability and Convergence Analysis

In this section, the stability and convergence properties of the μRNN are exactly analyzed. It is clear that φ is continuously differentiable. Thus φ is locally Lipschitz continuous in $\mathbb{R}^{n+m}$ with positive constant $\|\nabla\varphi\|$ where $\nabla\varphi$ is the Jacobian matrix for $\varphi(y)$. So, by Theorem 2 the solution $y(t)$, for $t \in [t_0, \tau)$ to the μRNN, for some $\tau > t_0$ is unique as $\tau \to \infty$.

Definition 2. ([25]) The equilibrium point y^* is Lyapunov stable if, for each $\epsilon > 0$, there is $\delta > 0$ such that if $\|y(t_0) - y^*\| < \delta$, then $\|y(t) - y^*\| < \epsilon$, for all $t \geq t_0$.

Definition 3. ([41]) A set G is an invariant set for a dynamic system if every system trajectory which starts from a point in G remains in G for all future times.

Theorem 4. (Local Invariant Set Theorem) Consider an autonomous system of the form $\dot{X} = f(X)$, with f continuous, and let $V(X)$ be a scalar function with continuous first partial derivatives. Assume that

- for some $l > 0$, the region Ω_l defined by $V(X) < l$ is bounded.
- $\frac{d}{dt} V(X) \leq 0$ for all X in Ω_l.

Let R be the set of all points within Ω_l where $\frac{d}{dt}V(X) = 0$, M be the largest invariant set in R. Then, every solution $X(t)$ originating in Ω_l tends to M as $t \to \infty$.

Proof. See [41]. □

Theorem 5. Let y^* be an equilibrium point for (5) and $D \subset \mathbb{R}^{n+m}$ be a domain containing y^*. Let $V : D \to \mathbb{R}$ be a continuously differentiable function such that

$$V(y^*) = 0, \;\; V(y) > 0, \;\; \text{in } D - \{y^*\}, \tag{11}$$

$$\frac{dV}{dt}(y) \leq 0, \;\; \text{in } D, \tag{12}$$

then, y^* is stable. Moreover, if

$$\frac{dV}{dt}(y) < 0, \;\; \text{in } D - \{y^*\},$$

then, y^* is asymptotically stable.

Proof. See [25]. □

Theorem 6. If $y = (X^T, \mu^T)^T \in \mathcal{M} \subset \mathbb{R}^{n+m}$, $X \in S_0$ and assume that

$$\mathcal{A} = A_f + \sum_{i=1}^{m} \frac{\mu_i^2}{2} A_{g_i} \succcurlyeq 0.$$

Then the Jacobian matrix $\nabla\varphi(y)$ of the mapping φ defined in (6) is a negative semi-definite matrix for all $y \in \mathcal{M}$.

Proof. It can be proved that the Jacobian matrix of φ is

$$\nabla\varphi = \begin{pmatrix} \left[-A_f - \sum_{i=1}^{m} \frac{\mu_i^2}{2} A_{g_i}\right]_{(n\times n)} & | & \left[-\nabla g^T(x).diag(\mu_1,\ldots,\mu_m)\right]_{(n\times m)} \\ - & - & - \\ [diag(\mu_1,\ldots,\mu_m).\nabla g(x)]_{(m\times n)} & | & [diag(g_1(x),\ldots,g_m(x))]_{(m\times m)} \end{pmatrix},$$

where $\nabla\varphi$ is an $(n+m)\times(n+m)$ matrix. $\nabla\varphi$ is exactly in the form of matrix $\mathcal{F}$ in Lemma 2. Since for any feasible point $y = (X^T, \mu^T)^T$ we have $g_i(X) \leq 0$, $i = 1,\ldots,m$ and using the assumption and Lemma 2 we obtain that $\nabla\varphi$ is a negative semi-definite matrix. □

Theorem 7. Let the assumptions of Theorem 6 hold. If $\Omega^* \subset \mathcal{M} \subset \mathbb{R}^{n+m}$ and S_0 is the feasible set of (1), then system (6) satisfies the following statements:

(i) Equilibrium points of (6) are stable in the sense of Lyapunov,

(ii) For all points $y(t_0) = (X_0^T, \mu_0^T)^T \in \mathcal{M}$ where $X_0 \in S_0$ the trajectory of $y(t)$ starting from $y(t_0)$ tends to Ω^* as $t \to \infty$.

(iii) For all $\hat{y} = (\hat{X}^T, \hat{\mu}^T)^T \in \Omega^*$ there exists a trajectory $y(t)$ with initial point $y(t_0) \in \mathcal{M}$ converges to $\hat{y}$, where $\hat{X}$ is a global optimal solution of problem (1).

Proof. (i). In this case, we prove that the equilibrium point is stable in the sense of Lyapunov. Consider the Lyapunov function $L : \mathcal{M} \to \mathbb{R}$ as

$$L(y) = \|y - \hat{y}\|^2, \tag{13}$$

where $\hat{y} \in \Omega^*$, and $y(t)$ is a trajectory obtained for system (6) starting from $y(t_0)$. Taking the time derivative from (13) and using (6), we have

$$\frac{dL(y)}{dt} = 2(y - \hat{y}).\frac{dy}{dt} = 2(y - \hat{y})\varphi(y).$$

Now by using the mean value theorem, there exists $\tilde{y}$ between y and $\hat{y}$, such that

$$\varphi(y) - \varphi(\hat{y}) = \nabla\varphi(\tilde{y})(y - \hat{y}).$$

By using Theorem 6 $\nabla\varphi(y)$ is negative definite, by multiplying both sides of the above equation by $(y - \hat{y})^T$, we have

$$(y - \hat{y})^T(\varphi(y) - \varphi(\hat{y})) = (y - \hat{y})^T\nabla\varphi(y)(y - \hat{y}).$$

Since the right side of the above equation is negative and $\varphi(\hat{y}) = 0$, we have

$$(y - \hat{y})^T\varphi(y) \leq 0 \Longrightarrow \frac{dL(y)}{dt} \leq 0.$$

(ii). To prove this part, we use Theorem 4. Note that system (6) is autonomous. The function $L : \mathcal{M} \to \mathbb{R}$, as stated in (13), is a scalar function with continuous first-order partial derivatives. For all $l > 0$, the following set

$$\Omega_l = \{y \in \mathcal{M} | L(y) \leq l\},$$

is bounded and for all $y \in inn(\Omega_l)$, we have $\frac{dL(y)}{dt} \leq 0$ where $inn(\Omega_l)$ is the interior of $\Omega(y)$. Now, Theorem 4 implies that every solution $y(t)$ of (6) starting from an arbitrary point belongs to $\mathcal{M}$, converges to a set of $M = \Omega^*$. It should be noted that in this discussion, the two sets R and M in Theorem 4 are the same as Ω^*.

(iii). If $y(t_0)$ is a feasible point, then the trajectory $y(t)$ obtained for the system of (6) starting from $y(t_0)$ cannot be unbounded, otherwise, we have $\lim_{k\to+\infty}\|y(t_k) - \hat{y}\|^2 = +\infty$ which leads to a contradiction with Lyapunov stability of system (6). As a result $\lim_{k\to+\infty}\|y(t_k) - \hat{y}\|^2 = 0$ or $M > 0$, if $\lim_{k\to+\infty}\|y(t_k) - \hat{y}\|^2 = 0$, then $\lim_{k\to\infty} y(t_k) = \hat{y}(t)$. If not, we have:

$$\lim_{k\to+\infty}\|y(t_k) - \hat{y}\|^2 = M > 0,$$

which implies that $\lim_{k\to+\infty} y(t_k) = \bar{y}$, where $\bar{y} \in \Omega^*$ and $\|\bar{y} - \hat{y}\|^2 = M$. Therefore, there is a trajectory $\{y(t_k)\}_{k=1}^{+\infty}$ that converges to an equilibrium point of system (6). According to Theorem 3, each equilibrium point of the system (6) is a global optimal solution of the problem (1). Therefore, the statement in item (iii) is true. □

In fact, all RNNs are autonomous.

Remark 2. In many problems, the attraction region of an equilibrium point is large, so the system (6) can be stable by starting from outside the feasible set.

5 Numerical Examples

In this section, some experiments are given to illustrate the efficiency and good performance of μRNN for solving optimization problems (1). The numerical testing was carried out on a Dell laptab (2.1 GHz, 2.00 GB of RAM) with the use of MATLAB (2008). The first three examples are for the non-convex problems and then the two examples are written for the convex problems. The numerical results of the non-convex examples are compared to the numerical results of Malk et al. [31] and convex examples are compared with Nazemi [36, 37].

Example 1. Consider the following non-convex quadratic optimization problem [31, 49].

$$\begin{array}{ll} \min & 8x_1x_2 + 3x_2^2 + 14x_1 + 12x_2 \\ s.t. & 18x_1^2 + 8x_2^2 + 2x_1 - 1 \leq 0, \\ & 13x_1^2 - 4x_1x_2 + 8x_2^2 + 4x_2 - 1 \leq 0, \\ & 5x_1^2 - 10x_1x_2 + 5x_2^2 + 16x_1 + 18x_2 - 1 \leq 0. \end{array} \tag{14}$$

By performing Network μRNN and starting from random initial points

$$y(t_0) = [rand, rand, rand, rand, rand\,]^T,$$

we obtain $X^* = (-0.21901076, -0.26801087)^T$ and $\mu^* = (2.00739048, 0.0001376, 0)^T$. From $\lambda_i = \frac{\mu_i^{*2}}{2}$ we have $\lambda^* = (2.014807306, 0, 0)^T$. We note that

$$\mathcal{A} = A_f + \frac{1}{2}\sum_{i=1}^{3}\mu_i^{*2}A_{g_i} = \begin{pmatrix} 72.5330630160 & 8 \\ 8 & 38.236916895 \end{pmatrix} \succ 0,$$

and $det(\mathcal{A}) = 2709.44 > 0$. In the system (6) a sufficiently large K could accelerate the μRNN. In other words, as the amount of K increases, the settling time of the system decreases significantly. Figure 2 presents the state trajectories of network μRNN with five random initial points and $K = 1$, $K = 1000$. According to the numerical results, it can be said that the convergence rate of networks μRNN and $M.RNN$ are equal.

Example 2. (Global solution for the CDT problem, [31]) Consider the following problem:

$$\begin{array}{ll} \min & f(d) = \frac{1}{2}d^TBd + b^Td \\ s.t. & \|A^Td + a\| \leq \theta, \\ & \|d\| \leq \delta, \end{array} \tag{15}$$

where $B \in S^n$, $A \in \mathbb{R}^{n\times m}(m \leq n)$, $b \in \mathbb{R}^n$, $a \in \mathbb{R}^m$, $\theta > 0$ and $\delta > 0$. The problem (15) comes from applying the successive quadratic programming method and the trust-region technique to minimize a general function $q(X)$ subject to $h(X) = 0$ (for the details

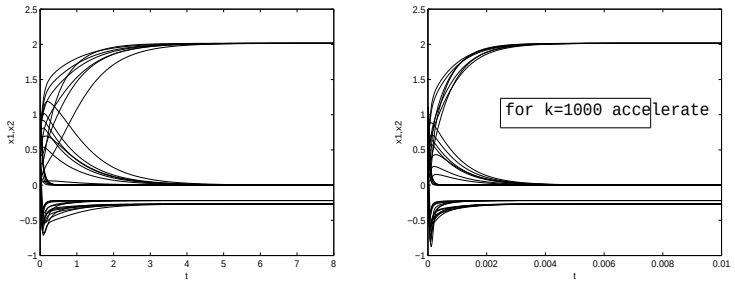

Figure 2: The transient behavior of the μRNN with different initial points for Example 1, $K = 1$ left side and $K = 1000$ right side.

see [31]). To solve the CDT problem (15) by the theory developed in this paper, we replace (15) by:

$$\begin{array}{ll} \min & f(d) = \frac{1}{2} d^T B d + b^T d \\ s.t. & g_1(d) = \|A^T d + a\|^2 - \theta^2 \leq 0, \\ & g_2(d) = \|d\|^2 - \delta^2 \leq 0.\delta, \end{array} \tag{16}$$

where for $n = m = 2$,

$$B = \begin{pmatrix} -2 & 0 \\ 0 & 2 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \; a = (0,-6)^T, \; b = (0,-6)^T, \; \delta = 5, \; \theta = 5,$$

$$H_f = \begin{pmatrix} B & b \\ b & 0 \end{pmatrix}, \; H_{g_1} = 2\begin{pmatrix} AA^T & Aa \\ (Aa)^T & \|a\|2 - \theta^2 \end{pmatrix}, \; H_{g_2} = \begin{pmatrix} I_n & 0 \\ 0 & -\delta^2 \end{pmatrix}.$$

By performing network μRNN and starting from random initial points, we obtain

$$d_I^* = \begin{pmatrix} 3.98163 \\ 3.00000 \end{pmatrix}, \; \mu_I^* = \begin{pmatrix} 1.82558 \\ 1.82558 \end{pmatrix}, \tag{17}$$

and

$$d_{II}^* = \begin{pmatrix} -3.98163 \\ 3.00000 \end{pmatrix}, \; \mu_{II}^* = \begin{pmatrix} 1.82558 \\ 1.82558 \end{pmatrix}, \tag{18}$$

are two different global optimal solutions for Example 2.

Note that

$$\mathcal{A} = A_f + \sum_{i=1}^{2} \frac{\mu_i^{*2}}{2} A_{g_i} = \begin{pmatrix} 4.6654 & 0 \\ 0 & 8.6654 \end{pmatrix} \succ 0.$$

Figure 3 presents the state trajectories of network μRNN with 2 random initial points and $K = 100$.

Example 3. Consider the following non-convex programming problem ([31]).

$$\begin{array}{ll} \min & -3x_1^2 + x_2^2 + \frac{3}{2}x_3^2 + 2x_4^2 + 3x_5^2 \\ s.t. & \frac{1}{4}(x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 14) \leq 0, \\ & \frac{1}{4}(x_1^2 + x_2^2 + x_3^2 + (x_4 - 3)^2 + x_5^2 - 17) \leq 0, \\ & -x_2x_3 - 0.5x_3^2 - 1.5x_4 + x_5^2 - 2.5 \leq 0, \\ & -2x_2x_3 + 0.5x_4^2 - 9x_5 \leq 0, \quad -x_2x_3 - 9x_5 \leq 0. \end{array} \tag{19}$$

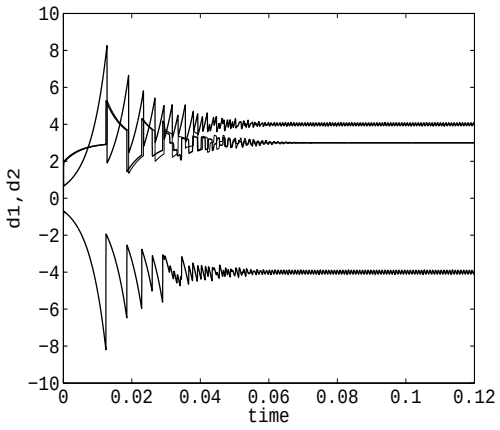

Figure 3: The transient behavior of the μRNN with different initial points for Example 2 for $K = 100$.

Using the network $M.RNN$, Malek and Hossinipour [31] obtained the optimal solutions as follows

$$X^* = (3.6051, 0.0000, 0.0000, 1.0000, 0.0556)^T,$$

$$X^{**} = (-3.6051, 0.0000, 0.0000, 1.0000, 0.0556)^T,$$

with

$$\lambda^* = \lambda^{**} = (5.2840, 6.7160, 0.0000, 0.0741, 0.0000)^T.$$

Now by using network μRNN, we obtain

$$x_I^* = \begin{pmatrix} 3.606758 \\ 0 \\ 0 \\ 0.999630707 \\ 0.05558959 \end{pmatrix}, \quad \mu_I^* = \begin{pmatrix} 3.249422 \\ 3.663398 \\ 0 \\ 0.3848944 \\ 0 \end{pmatrix}, \tag{20}$$

and

$$x_{II}^* = \begin{pmatrix} -3.605471 \\ 0 \\ 0 \\ 0.999979 \\ 0.0555559 \end{pmatrix}, \quad \mu_{II}^* = \begin{pmatrix} 3.251679 \\ 3.666151 \\ 0 \\ 0.384872 \\ 0 \end{pmatrix}. \tag{21}$$

Moreover, we get

$$\mathcal{A} = \begin{pmatrix} 18.014 & 0 & 0 & 0 & 0 \\ 0 & 26.014 & -0.1481 & 0 & 0 \\ 0 & -0.1481 & 27.0140 & 0 & 0 \\ 0 & 0 & 0 & 28.0881 & 0 \\ 0 & 0 & 0 & 0 & 30.0140 \end{pmatrix}. \tag{22}$$

In this problem, since H_f and H_{g_i}, $i = 1,\ldots,m$ are Z-matrices, we can conclude by Theorem 1 X_I^* and X_{II}^* are two different global solutions of this problem. Figures 4 and 5 present the state trajectories of network μRNN with two random initial points and $K = 1$, $K = 300$, $K = 1300$. According to the numerical results, it can be said that the convergence rates of network μRNN are better than the network $M.RNN$.

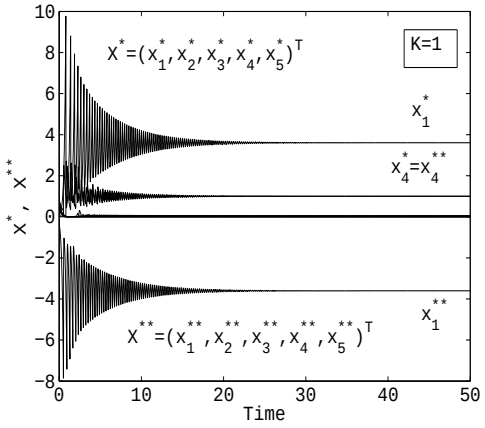

Figure 4: The transient behavior of the μRNN with different initial points for Example 3 for $K = 1$.

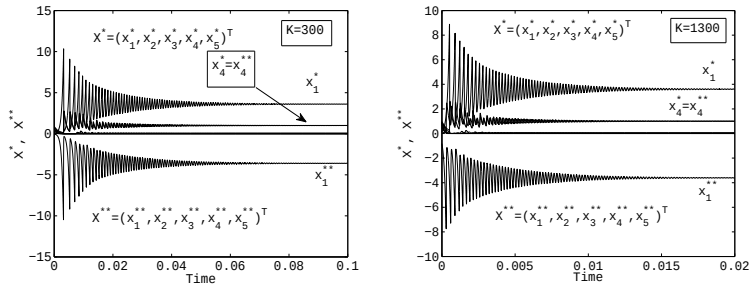

Figure 5: The transient behavior of the μRNN with different initial points for Example 3, $K = 300$ left side and $K = 1300$ right side.

To check the μRNN performance, we want to solve two examples for convex problems.

Example 4. Consider the following convex nonlinear optimization problem [36].

$$\begin{array}{ll} \min & x_1^2 + 2x_2^2 + 2x_1x_2 - 10x_1 - 12x_2 \\ s.t. & z_1 + 3x_2 \leq 8, \\ & x_1^2 + x_2^2 + 2x_1 - 2x_2 \leq 3. \end{array} \tag{23}$$

The exact solution is $X^* = (1,2)^T$. By performing the network μRNN and starting from 5 random initial points we get

$$x^* = \begin{pmatrix} 1.000075 \\ 2.000020 \end{pmatrix}, \quad \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} 0.001533 \\ 1.414195 \end{pmatrix}, \tag{24}$$

and note that

$$\mathcal{A} = A_f + \lambda_1 Ag_1 + \lambda_2 Ag_2 = \begin{pmatrix} 3.99994 & 2.0000 \\ 2.0000 & 5.999947 \end{pmatrix} \succ 0. \tag{25}$$

Figure 6 displays the transient behavior based on the network μRNN with 5 random initial points. All trajectories of the network converge to $X^* = (1,2)^T$. Moreover, when the initial point is chosen as an infeasible point, the trajectory of the network μRNN still converges to X^*.

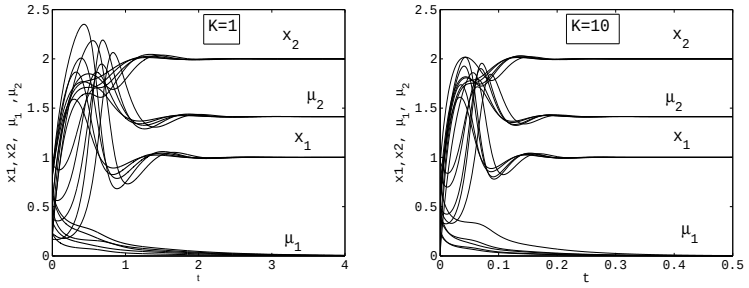

Figure 6: Transient behaviors of $y(t) = (X^T, \mu^T)^T$ of network μRNN with 5 various initial points in Example 4 for $K = 1$ and $K = 10$.

Example 5. Consider the following convex nonlinear optimization problem [37]:

$$\begin{array}{ll} \min & 10x_1^2 + 2x_2^2 + 2x_3^2 - 2(x_1x_2 + 3x_1x_3 - x_2x_3) \\ s.t. & -1 \le x_2 - x_1 \le 0, \\ & -1 \le x_3 - 3x_1 \le 1, \\ & 1 \le x_2 + x_3 \le 2. \end{array} \tag{26}$$

The first constraint is equivalent to $(x_2 - x_1)(x_2 - x_1 + 1) \le 0$. Similarly, the constraints of the problem (26) are equivalent to the following constraints.

$$\begin{cases} x_1^2 + x_2^2 - 2x_1x_2 + x_2 - x_1 \le 0, \\ 9x_1^2 + x_3^2 - 6x_1x_3 - 1 \le 0, \\ x_2^2 + x_3^2 + 2x_2x_3 - 3x_2 - 3x_3 + 2 \le 0. \end{cases}$$

Since the problem is convex, the unique exact solution is $X^* = (\frac{1}{4}, \frac{1}{4}, \frac{3}{4})^T$. By performing the network μRNN and starting from 4 random initial points we obtain

$$x^* = \begin{pmatrix} 0.250452 \\ 0.248640 \\ 0.75135 \end{pmatrix}, \quad \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{pmatrix} = \begin{pmatrix} 0.146195 \\ 0 \\ 1.999987 \end{pmatrix}. \tag{27}$$

Note that

$$\mathcal{A} = A_f + \sum_{i=1}^{3} \lambda_i Ag_i = \begin{pmatrix} 5.02 & -0.50 & -1.50 \\ 2.00 & 5.99 & 2.01 \\ -1.50 & 2.01 & 2.51 \end{pmatrix} \succ 0, \tag{28}$$

and $det(\mathcal{A}) = 8.128$. Figures 7 and 8 show that the trajectories of the network μRNN to solve the above problem with 4 random initial points and $K = 15, 120, 1000$, converge to the optimal solution of this problem. It is seen that the proposed network converges to the exact solution X^* independent of the way that we may choose the starting points.

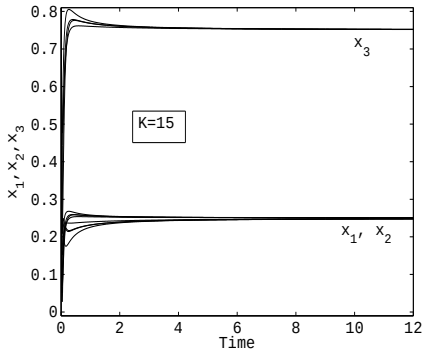

Figure 7: The transient behavior of the μRNN with 4 initial points for Example 5 for $K = 15$.

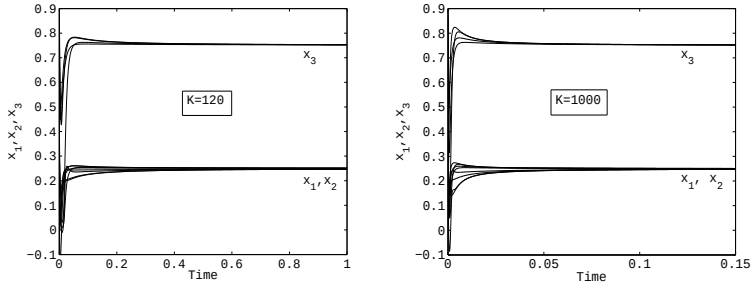

Figure 8: The transient behavior of the μRNN with 4 initial points for Example 5, $K = 120$ left side and $K = 1000$ right side.

6 Conclusion

In this paper, we presented a recurrent neural network (μRNN) for solving non-convex quadratic problems based on the Lyapunov theory. This network is a modified $M.RNN$ and has a simpler structure compared to that. The capability of this network is equal to and sometimes better than $M.RNN$. Finally, to show the efficiency of the proposed network, some numerical examples (convex and non-convex) were presented.

References

[1] Bacciotti, A. (1992). "Local stabilizability of nonlinear control systems", Advances in Mathematics for Applied Sciences.

[2] Bazaraa, M.S., Shetty, C.M. (1990). "Nonlinear programming theory and algorithms", Wiley and Sons, New York.

[3] Bertsekas, D.P. (1989). "Parallel and distributed numerical methods", Prentice-Hall, Englewood, Cliffs, NJ.

[4] Best Michael, J. (2017). "Quadratic programming with computer programs", Advances in Applied Mathematics, University of Waterloo, Canada, CRC Press.

[5] Beyer, D., Ogier, R. (2000). "Tabu learning: A neural networks search method for solving non-convex optimization problems", IEEE International Joint Conference Neural Networks 2, 953-961.

[6] Bian, W., Chen, X. (2013). "Worst-case complexity of smoothing quadratic regularization methods for non-Lipschitz optimization", SIAM, Journal of Optimization, 23 (3), 1718-1741.

[7] Bian, W., Chen, X., Ye, Y. (2014). "Complexity analysis of interior point algorithms for non-Lipschitz and non-convex minimization", Mathematical Programming, 149(1-2), 301-327.

[8] Boob, D.P. (2020). "Convex and structured non-convex optimization for modern machine learning: Complexity and algorithms", Georgia Institute of Technology.

[9] Carmon, Y., Duchi, J.C. (2020). "First-order methods for non-convex quadratic minimization", SIAM Review, 62(2), 395-436.

[10] Chen, X., Womersley, R., Ye, J. (2011). "Minimizing the condition number of a Gram matrix", SIAM Journal on Optimization, 21, 127-148.

[11] Chicone, C. (2006). "Ordinary differential equations with applications"; Second edition, Springer-Verlag, New York.

[12] Cui, Y., Chang, T. H., Hong, M., Pang, J.S. (2020). "A study of piecewise linear-quadratic programs", Journal of Optimization Theory and Applications, 1-31.

[13] Effati, S., Mansoori, A., Eshaghnezhad, M. (2015). "A projection neural network for solving bilinear programming problems", Neurocomputing, 1-20.

[14] Effati, S., Ranjbar, M. (2011). "A novel recurrent nonlinear neural network for solving quadratic programming problems", Applied Mathematical Modelling, 33, 1688-1695.

[15] Eshaghnezhad, M., Effati, S., Mansoori, A. (2016). "A neurodynamic model to solve nonlinear pseudo-monotone projection equation and its applications", IEEE Transactions on Cybernetics, 1-13.

[16] Gao, X.B., Liao, L.Z., Xue, W. (2004). "A neural network for a class of convex quadratic minimax problems with constraints", IEEE Transactions on Neural Networks, 15(3), 622-628.

[17] Hopfield, J.J., Tank, D. (1985). "Neural computation of decisions in optimization problem", Biod. Cybern., 52, 141-152.

[18] Horn, R.A., Johnson, C.R. (1990). "Matrix Analysis", Cambridge University Press.

[19] Huan, L., Fang, C., Zhouchen, L. (2020). "Accelerated first-order optimization algorithms for machine learning", Proceedings of the IEEE, doi: 10.1109/JPROC.2020.3007634.

[20] Huan, L., Lin, Z. (2019). "Provable accelerated gradient method for non-convex low-rank optimization", Machin Learning.

[21] Huang, F., Gu, B., Huo, Z., Xhen, S., Huang, H. (2019). "Faster gradient-free proximal stochastic methods for non-convex nonsmooth optimization", The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19), 1503-1510.

[22] Jeyakumar, V., Lee, G.M., Li, G.Y. (2009). "Alternative theorems for quadratic inequality systems and global quadratic optimization", SIAM Journal on Optimization, 20(2), 983-1001.

[23] Jeyakumar, V., Rubinov, A.M., Wu, Z.Y. (2007). "Non-convex quadratic minimization problems with quadratic constraints: Global optimality conditions", Mathematical Programming, series, A 110, 521-541.

[24] Jeyakumar V., Srisatkunarajah S. (2009). "Lagrange multiplier necessary condition for global optimality for non-convex minimization over a quadratic constraint via S-lemma", Optimization Letters, 3, 23-33.

[25] Khalil, H.K. (2002). "Nonlinear systems", Prentice Hall, Third edition.

[26] Kong, W., Melo, J.G., Monteiro, R.D.C. (2019). "An efficient adaptive accelerated inexact proximal point method for solving linearly constrained non-convex", Composite Problems, 1-28.

[27] Leung, M.F., Wang, J. (2019). "Minimax and bi-objective portfolio selection based on collaborative neurodynamic optimization", IEEE Transactions on Neural Networks and Learning Systems, 1-12.

[28] Liu, S., Jiang, H., Zhang, L., Mei, X. (2020). "A neurodynamic optimization approach for complex-variables programming problem", Neural Networks, 129, 280-287.

[29] Lu, S. (2018). "First-Order methods of solving non-convex optimization problems: Algorithms, convergence, and optimality", Electrical and Electronics Commons.

[30] Luenberger, D.G. (1948). "Introduction to Linear and Nonlinear Programming", Reading MA: Addison-Wesley.

[31] Malek, A., Hosseinipour-Mahani, N. (2015). "Solving a class of non-convex quadratic problems based on generalized KKT conditions and neurodynamic optimization technique", Kybernetika, 51, 890-908.

[32] Mansoori, A., Effati, S. (2019). "An efficient neurodynamic model to solve nonlinear programming problems with fuzzy parameters", Neurocomputing, 334, 125-133.

[33] Mansoori, A., Effati, S. (2019). "Parametric NCP-based recurrent neural network model: A new strategy to solve fuzzy non-convex optimization problems", IEEE Transactions on Systems, Man, and Cybernetics: Systems, 1-10.

[34] Modarres, F., Hassan, M.A., Leong, W.J. (2011). "A symmetric rank-one method based on extra updating techniques for unconstrained optimization", Computers and Mathematics with Applications, 62, 392-400.

[35] Nasiri, J., Modarres Khiyabani, F. (2018). "A whale optimization algorithm (WOA) approach for clustering", Cogent Mathematics and Statistics, doi.org/10.1080/25742558.2018/1483656.

[36] Nazemi, A.R. (2012). "A dynamic system model for solving convex nonlinear optimization problems", Communications in Nonlinear Science and Numerical Simulation, 17, 1696-1705.

[37] Nazemi, A.R. (2014). "A neural network model for solving convex quadratic programming problems with some applications problems", Engineering Applications of Artificial intelligence, 32, 54-62.

[38] Pant, H., Soman Jayadeva, S., Bhaya, A. (2020). "Neurodynamical classifiers with low model complexity", Neural Networks, 132, 405-415.

[39] Park, S., Jung, S.H., Pardalos, P.M. (2020). "Combining stochastic adaptive cubic regularization with negative curvature for non-convex", Journal of Optimization Theory and applications, 184 (3), 953-071.

[40] Rudnick-Cohen, E., Herrmann, J.W., Azarm, S. (2020). "Non-convex feasibility robust optimization via scenario generation and local refinement", Journal of Mechanical Design, 142 (5), 1-10.

[41] Slotine, J.J.E., Li, W. (1990). "Applied Nonlinear Control", Wiley and Sons, New York.

[42] Strekalovsky, A.S. (2018). "On non-convex optimization problems with D. C. equality and inequality constraints", IFAC papers online, 51-32, 895-900.

[43] Strekalovsky, A. (2019). "Nonconvex optimization: From global optimality conditions to numerical methods", AIP Conference Proceedings 2070, 020015, 1-4.

[44] Tank, D.W., Hopfield, J.J. (1986). "Simple neural optimization networks: On A/D converter, signal decision circuit and a linear programming circuit", IEEE Transactions on Circuits and Systems, 33, 533-541.

[45] Tian, Y., Lu, C. (2011). "Nonconvex quadratic formulations and solvable conditions for mixed integer quadratic programming problems", Journal of Industrial and Management Optimization, 7 (4), 1027-1039.

[46] Valizadeh Oghani, A., Khiabani, F. M., Farahmand, F.H. (2020). "Data envelopment analysis technique to measure the management ability: Evidence from Iran cement industry", Cogent Business and Management, doi.org/10.1080/23311975.2020.1801960.

[47] Xu, C., Chai, Y., Qin, S., Wang, Z., Feng, J. (2020). "A neurodynamic approach to nonsmooth constrained pseudo convex optimization problem", Neural Networks, 124, 180-192.

[48] Xue, X., Bian, W. (2007). "A project neural network for solving degenerate convex quadratic program", Neurocomputing, 70, 2449-2459.

[49] Yan, Y. (2014). "A new nonlinear neural network for solving quadratic programming problems", Springer International Publishing Switzerland, 347-357.

[50] Yang, Y., Cao, J., Xu, X., Liu, J. (2012). "A generalized neural network for solving a class of minimax optimization problems with linear constraints", Applied Mathematics and Computation, 218, 7528-7537.

How to Cite this Article:
Mohammadsalahi, K., Modarres Khiyabani, F., Azarmir Shotorbani N., (2022). "A new optimization method based on dynamic neural networks for solving non-convex quadratic constrained optimization problems", Control and Optimization in Applied Mathematics, 7(2): 35-52. doi: 10.30473/coam.2022.64268.1206