Journal of Nonlinear and Convex Analysis
Volume 26, Number 9, 2025, 2757–2776

Yokohama Publishers
ISSN 1880-5221 ONLINE JOURNAL

CONVERGENCE OF A DELAYED NEURAL NETWORK FOR A SECOND-ORDER CONE PROGRAMMING AND SIMULATIONS

JIE ZHANG, JIANI YANG, AND YIFEI WANG

Abstract. In this study, a delayed neural network (DNN) framework tailored specifically for solving second-order cone programming (SOCP) problems is presented. The cornerstone of this method is the integration of optimality conditions with projection operators, which serve as fundamental building blocks for the network's design. Initially, the problem of identifying optimal solutions within the realm of SOCP is reframed as the quest for equilibrium points within the proposed DNN framework. Subsequently, under specified conditions, we rigorously prove existence and stability of these equilibrium points within DNN. This transformation and subsequent analysis pave the way for utilizing the DNN as an effective tool for solving SOCP problems. Finally, we performed several numerical experiments, including those for solving the linear and quadratic SOCP problems, as well as the optimal grasping force multi-finger manipulator problem. Simulations were subsequently conducted, confirming the feasibility and effectiveness of the DNN.

1. Introduction

The SOCP broadens the scope of conic optimization by incorporating constraints defined by second-order cones. This expansion has enabled its application in diverse fields such as control theory, where it aids in the design of robust controllers; signal processing, where it facilitates efficient signal recovery and filtering; and machine learning, where it contributes to the development of advanced algorithms for optimization and decision-making. Many familiar optimization problems, such as distributionally robust optimizations and convex quadratic programming (QP), can be transformed into SOCP. The superiority of SOCP has made it a hot research topic for scholars, and the research results on SOCP models and numerical methods have deepened and enriched the field, see [10] and [1] and reference therein. Recently, there are many significant progress in the understanding and solution of SOCP problems. Fukuda et al. [4] formulated a new second-order necessary optimality criteria for nonlinear SOCP problems, contingent upon the fulfillment of Robinson's constraint qualification. Their findings hold significance for the advancement of both first and second-order algorithmic approaches. Meanwhile, Liang et al. [8] embraced a numerical Method for SOCP and showcasing competitive robustness of solvers like Mosek and SDPT3. Andreani et al. [2] delved into the interplay between optimality conditions in nonlinear programming and SOCP, introducing an

2020 *Mathematics Subject Classification.* 90C33, 34K35, 93D20, 65K10.

Key words and phrases. SOCP, delayed neural network, metric projection, existence and uniqueness, exponential convergence.

The research presented in this study was financially supported by the National Natural Science Foundation of China under Project Grant Numbers 12171219 and 61877032.

augmented Lagrangian approach characterized by its global convergence properties. While significant strides have been made, there remains a need for more efficient algorithms. Additionally, the integration of SOCP with other optimization techniques and its application in emerging fields such as artificial intelligence and big data analytics present fertile grounds for future research.

While traditional approaches to tackling SOCP, encompassing interior-point methods and smoothing Newton methods, have undergone rigorous examination, they may falter in real-time applications owing to their intricate computational demands. This underscores the need for more efficient strategies tailored for such time-sensitive scenarios. In response, researchers have turned to neural network approaches, which offer potential advantages in terms of computational efficiency and hardware implementability. A notable contribution in this domain involves the conception of a neural network architecture grounded in the generalized Fischer-Burmeister function, empirically proven efficient in resolving nonlinear convex programs constrained by second-order cones [11]. This innovative model exhibits enhanced stability and expedited convergence rates, outperforming traditional optimization techniques in these key metrics. Further advancements have been made by proposing a projected-based neural network method that simplifies the SOCP problem into an equivalent projection equation, leading to reduced computational time and guaranteed convergence to the exact solution [22]. This approach leverages the properties of second-order cone projection, offering a stable and efficient solution to SOCP problems. In another significant development, a new neural network was introduced to solve SOC constrained variational inequality (SOCCVI) problem by first transforming it into a convex SOCP task [13]. The presented model, integrating a smoothing approach with a proficient neural network architecture, has been empirically validated to demonstrate Lyapunov stability and achieve global convergence towards an optimal solution. Furthermore, an innovative collaborative neuro-dynamic paradigm has been devised to tackle convex SOCP challenges, notably in the context of multi-fingered robotic manipulators, as demonstrated in [12]. More recently, a novel neural network framework has emerged, targeted at resolving SOCCVI, with a promising application in enhancing the capabilities of multi-fingered robotic hands, as outlined in [14]. This framework simplifies the problem by reducing the state variables and offers a model with lower structural complexity and weaker convergence conditions. Lastly, a projection neural network has been introduced to address SOCCVI problems, which achieves exponential stability under second-order sufficient conditions [18].

Time delay neural networks have advantages in addressing certain issues, as they can effectively capture delay phenomena within systems, thereby more clearly describing the complexity of the transmission process. In contrast, traditional neural networks may not accurately depict these complex dynamic behaviors. In recent years, researchers have proposed several types of DNN to solve optimization problems. Over the past two decades, there has been significant progress in employing DNNs to address various optimization problems. These networks have demonstrated their effectiveness in tackling complex and computationally intensive tasks across different fields, including engineering, economics, and applied sciences. One of the works in this area is the development of the DNN for solving QPs, as discussed

in [6]. Subsequent research has focused on enhancing the stability and convergence properties of DNN. For instance, the delayed projection neural network presented in [21] has been proven to be globally exponentially stable, offering a promising approach to solving a real-time convex QP. The introduction of time delays in neural networks, as in [7, 9, 23], has further enriched the methodology, allowing for the modeling of realistic scenarios where signal transmission and processing are subject to delays. These studies have shown that with appropriate delays. More recent contributions, such as the work in [15], [17] and [20], have extended the application of DNNs to convex QP with inequality constraints, which have been proven to possess global exponential stability, ensuring the practicality and reliability of DNNs in solving optimization problems. The research has also explored the impact of variable time delays on the performance of DNNs, as presented in [19]. However, no DNN method for solving SOCP problems has emerged yet.

Motivated by the previously discussed findings, this study presents a DNN designed to address a specific set of SOCP. To our fullest understanding, this marks the inaugural instance where a DNN has been employed to tackle the SOCP. Unlike traditional neural networks that tackle SOCP, such as those referenced in [14], [12], [18], this DNN accounts for the intrinsic time lags encountered in real-world scenarios, thereby enhancing its practical relevance. Additionally, in contrast to standard projection neural networks utilized for SOCP, the newly proposed projection DNN demands fewer state variables, as the multipliers from the equality constraints are omitted in the projection formula. By extending the technique of variation of constants, we establish that this model is both Lyapunov stable and globally convergent. The efficiency of this novel DNN is further substantiated through simulation outcomes on various test cases.

The organization of the subsequent sections in this paper is detailed as follows: Section II introduces a novel DNN, grounded in the relevant KKT conditions and a projection operator. Section III delves into the existence and uniqueness of the DNN's continuous solutions. The stability of the DNN's equilibrium points and the exponential convergence of its states are substantiated. Section IV showcases numerical examples and conducts simulations to validate the DNN's efficacy. In addition, a comparison with the existing model further highlights the superior efficiency of the proposed DNN in addressing the specific problem. Conclusively, Section V encapsulates the collective findings and contributions of this research endeavor.

2. The delayed neural network formulation

In this paper, the SOCP is specifically expressed as:

$$\begin{aligned} \min \quad & \frac{1}{2}x^\top Gx + c^\top x \\ \text{s. t.} \quad & Ax - b = 0, \\ & x \in K, \end{aligned} \tag{2.1}$$

where $x \in \mathbb{R}^n$, $G \in \mathbb{R}^{n\times n}$ is a symmetric positive semidefinite matrix, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^l$,$A \in \mathbb{R}^{l\times n}$ is of full row rank, the set K is expressed as $K = K^{m_1} \times K^{m_2} \times \cdots \times K^{m_p}$

with $m_i \geq 1$, $m_1 + m_2 + \cdots + m_p = n$, $i = 1, \cdots, p$. Each K^{m_i}, $i = 1, \ldots, p$ is a m_i-dimensional second-order cone defined as

$$K^{m_i} = \left\{ (x_{i1}, \cdots, x_{im_i})^T \in \mathbb{R}^{m_i} \mid \|(x_{i2}, \cdots, x_{im_i})\| \leq x_{i1} \right\},$$

$\|\cdot\|$ represents the Euclidean norm of a vector. It is worth noting that K^1 is defined as a set of nonnegative real numbers. Then in the case when $p = n$ and $m_1 = m_2 = \cdots = m_p = 1$, K is just $\mathbb{R}^n_+$.

Additionally, in this paper, we suppose that the optimal solution for (2.1) is unique and the feasible domain is not empty. Next we investigate the equivalent forms of the SOCP (2.1). Let Lagrange function be defined as:

$$L(x, v, u) = \frac{1}{2} x^T G x + c^T x + (Ax - b)^T v + u^T x.$$

Then by Theorem 3.6 in [3], is a optimal solution of (2.1) if and only if there exist Lagrange multipliers $v^* \in \mathbb{R}^l$, $u^* \in K^-$ such that the following KKT condition holds at $(x^*, v^*, u^*)^T$:

x^* is an optimal solution of (2.1) precisely when there exist Lagrange multipliers $v^* \in \mathbb{R}^l$, $u^* \in K^-$ satisfying the KKT condition at the point $(x^*, v^*, u^*)^T$:

$$\begin{cases} u^* \in K^-, x^* \in K, \langle u^*, x^* \rangle = 0, \\ Qx^* + c + A^T v^* + u^* = 0, \\ Ax^* - b = 0, \end{cases} \tag{2.2}$$

where K^- is the polar cone of K.

Notice that $u^* \in K^-, x^* \in K, \langle u^*, x^* \rangle = 0$ means that for any $\alpha > 0$, $\Pi_{K^-}(u^* + \alpha x^*) = u^*$, which is equivalent to $\Pi_K(x^* + \alpha u^*) = x^*$, where for a convex set $\Omega, \Pi_\Omega(\cdot)$ is the projection operator defined by $\Pi_\Omega(y) = \arg\min_{w \in \Omega} \|y - w\|$. The detailed description of $\Pi_K(\cdot)$ can be found in [16] through spectral decomposition.

To simplify the structure of the equation system (2.2), we provide the following theorem.

Theorem 2.1. *[KKT Characterization for SOCP] Let x^* be a feasible solution of the second-order cone program (2.1). Then x^* is an optimal solution if and only if there exist Lagrange multipliers $v^* \in \mathbb{R}^l$, $u^* \in K^-$, such that the KKT conditions are satisfied at the point $(x^*, v^*, u^*)^T$:*

$$\begin{cases} (E - B)(Gx^* + c + u^*) + Q(Ax^* - b) = 0, \\ \Pi_{K^-}\left[u^* + \alpha\left(x^* - \sigma\left((E - B)(Gx^* + c + u^*) + Q(Ax^* - b)\right)\right)\right] - u^* = 0, \end{cases} \tag{2.3}$$

where $B = A^T(AA^T)^{-1}A$, $Q = A^T(AA^T)^{-1}$, and E is the identity matrix. The constants satisfy $\alpha > 0$, $\sigma > 0$.

Proof. **Sufficiency.** Let x^* be an optimal solution to SOCP (2.1), in what follows we know from (2.2) that there exist $v^* \in \mathbb{R}^l$, $u^* \in K^-$ such that

$$x^* \in K, \quad Ax^* - b = 0, \quad (u^*)^T x^* = 0 \tag{2.4}$$

and

$$Gx^* + c + A^T v^* + u^* = 0.$$

Then it holds that

$$-A(Gx^* + c + A^T v^* + u^*) = 0. \tag{2.5}$$

Combining (2.4) and (2.5), We can further obtain

$$Ax^* - A(Gx^* + c + u^*) - AA^T v^* = b.$$

Thus we have

$$\begin{aligned} A^T v^* &= A^{-1}(Ax^* - b - A(Gx^* + c + u^*)) \\ &= A^T(AA^T)^{-1}(Ax^* - b) - A^T(AA^T)^{-1}A(Gx^* + c + u^*). \end{aligned} \tag{2.6}$$

Substituting (2.6) into $Gx^* + c + A^T v^* + u^* = 0$, then it holds that

$$Gx^* + c + A^T(AA^T)^{-1}(Ax^* - b) - A^T(AA^T)^{-1}A(Gx^* + c + u^*) + u^* = 0.$$

i.e.,

$$(I - A^T(AA^T)^{-1}A)(Gx^* + c + u^*) + A^T(AA^T)^{-1}(Ax^* - b) = 0.$$

Then we have

$$(E - B)(Gx^* + c + u^*) + Q(Ax^* - b) = 0. \tag{2.7}$$

Thus, the first equation in expression (2.3) holds.

Next, according to properties of the projection operator, $x^* \in K, u^* \in K^-$, $(u^*)^T x^* = 0$ means

$$\Pi_{K^-}[u^* + \alpha x^*] - u^* = 0. \tag{2.8}$$

Combining (2.7) with (2.8), we have

$$\Pi_{K^-}\left[u^* + \alpha\left((x^* - \sigma[(E - B)(Gx^* + c + u^*) + Q(Ax^* - b)]\right)\right] = u^*.$$

The second equation in (2.3) holds.

Necessity. Assume that u^* satisfies (2.3). then we have

$$A(E - B)(Gx^* + c + u^*) + AQ(Ax^* - b) = 0.$$

Notice that $A(E-B) = 0$ and $AQ = E$, so $(E-B)(Gx^*+c+u^*) = 0$, $Ax^* - b = 0$. Let

$$v^* = -(AA^T)^{-1}A(Gx^* + c + u^*),$$

then

$$Gx^* + c + A^T v^* + u^* = (E - B)(Gx^* + c + u^*).$$

Therefore, the existence of (x^*, v^*, u^*) makes the KKT condition (2.2) valid. □

Next, we will transform the above equation system (2.3) into a single projection equation:

Theorem 2.2. *The vector x^* constitutes the optimal solution to the problem (2.1) precisely when there exists a vector u^* such that the concatenated vector $y^* := (x^*, u^*)$ fulfills the projection equation:*

$$y^* = \Pi_C\left[y^* - \alpha(Hy^* + z)\right],$$

where $\alpha > 0$ is a constant, $C := \mathbb{R}^n \times K^-$ and

$$H = \begin{pmatrix} (E - B)G + B & (E - B) \\ \sigma[(E - B)G + B] & -\sigma[E - B] \end{pmatrix}, \quad z = \begin{pmatrix} (E - B)c - Qb \\ \sigma[(E - B)c - Qb] \end{pmatrix}.$$

Proof. According to the projection concept, the set of equations referenced by (2.3) admits a representation in the subsequent format:

$$\Pi_{\mathbb{R}^n}\left[x^* - \alpha((E-B)(Gx^*+c+u^*)+Q(Ax^*-b))\right] - x^* = 0$$

and

$$\Pi_{K^-}\left[u^* + \alpha((x^* - \sigma((E-B)(Gx+c+u^*)+Q(Ax^*-b)))\right] - u^* = 0.$$

Therefore

$$\begin{aligned} x^* &= \Pi_{\mathbb{R}^n}\left[x^* - \alpha((E-B)(Gx^*+c+u^*)+Q(Ax^*-b))\right] \\ &= \Pi_{\mathbb{R}^n}\left[\begin{pmatrix} E & 0\end{pmatrix}\begin{pmatrix} x^* \\ u^*\end{pmatrix}\right. \\ &\qquad \left. -\alpha\left(\begin{pmatrix}(E-B)G+B & (E-B)\end{pmatrix}\begin{pmatrix} x^* \\ u^*\end{pmatrix} + \begin{pmatrix}(E-B)c-Qb\end{pmatrix}\right)\right]. \end{aligned}$$

On the other hand,

$$\begin{aligned} u^* &= \Pi_{K^-}\left[u^* + \alpha(x^* - \sigma((E-B)(Gx^*+c+u^*)+Q(Ax^*-b)))\right] \\ &= \Pi_{K^-}\left[\begin{pmatrix} 0 & E\end{pmatrix}\begin{pmatrix} x^* \\ u^*\end{pmatrix}\right. \\ &\qquad \left. -\alpha\left(\begin{pmatrix}\sigma\left[(E-B)G+B\right]-E & \sigma(E-B)\end{pmatrix}\begin{pmatrix} x^* \\ u^*\end{pmatrix} + \begin{pmatrix}\sigma\left[(E-B)c-Qb\right]\end{pmatrix}\right)\right]. \end{aligned}$$

Combining the above two equations, it can be concluded that

$$\begin{aligned} \begin{pmatrix} x^* \\ u^*\end{pmatrix} &= \Pi_C\left[\begin{pmatrix} x^* \\ u^*\end{pmatrix}\right. \\ &\left. -\alpha\left(\begin{pmatrix}(E-B)G+B & E-B \\ \sigma[(E-B)G+B]-E & \sigma(E-B)\end{pmatrix}\begin{pmatrix} x^* \\ u^*\end{pmatrix} + \begin{pmatrix}(E-B)c-Qb \\ \sigma[(E-B)c-Qb]\end{pmatrix}\right)\right]. \end{aligned}$$

Let

$$H = \begin{pmatrix}(E-B)G+B & E-B \\ \sigma[(E-B)G+B]-E & \sigma(E-B)\end{pmatrix}, \quad z = \begin{pmatrix}(E-B)c-Qb \\ \sigma[(E-B)c-Qb]\end{pmatrix}.$$

Then

$$y^* = \Pi_C[y^* - \alpha(Hy^*+z)].$$

□

Based on the Theorem 2.2, we propose a new DNN to solve problem (2.1), namely

$$\begin{cases} \frac{dy(t)}{dt} = (\gamma-1)\Pi_C\left[y(t-r) - \alpha(Hy(t-r)+z)\right] \\ \qquad + \Pi_C\left[y(t) - \alpha(Hy(t)+z)\right] - \gamma y(t), \\ y(t) = \varphi(t), \quad t \in [-r, 0], \end{cases} \tag{2.9}$$

in which $\alpha > 0, \gamma > 0$, $r > 0$ represents time delay, $\varphi(t)$ is continues on $[-r, 0], C = \mathbb{R}^n \times K^-$.

3. Existence and stability

In this section, we will discuss the existence and stability of solutions to DNN (2.9) for SOCP problems.

We at first give a lemma to show the property of projection operator.

Lemma 3.1. *For a closed convex set $\Omega \subseteq \mathbb{R}^n$. We have*

$$\|\Pi_\Omega(z) - \Pi_\Omega(z')\| \leq \|z - z'\|$$

holds for any $z, z' \in \mathbb{R}^n$.

We next give the existence of solutions to DNN (2.9).

Theorem 3.2. *For every function φ that belongs to the space $\varphi \in C([-r, 0], \mathbb{R}^{2n})$, there exists precisely one continuous function $y(t)$ that serves as a solution to the DNN (2.9) throughout the entire global time interval $[0, +\infty)$.*

Proof. Let

$$\begin{aligned} F(y_t) \quad &= -\gamma y(t) + (\gamma - 1)\Pi_C \left[y(t - r) - \alpha(Hy(t - r) + z)\right] \\ &\quad + \Pi_C \left[y(t) - \alpha(Hy(t) + z)\right], \end{aligned}$$

where $y_t = y(t + \theta), -r \leq \theta \leq 0$. Define $\| \varphi \|_r = \sup_{-r \leq t \leq 0} \|\varphi(t)\|$. By Lemma 3.1, for any $\varphi, \varphi' \in C([-r, 0], \mathbb{R}^{2n})$,

$$\begin{aligned} &\|F(\varphi) - F(\varphi')\| \\ &\leq 2 \| \varphi - \varphi' \|_r + (\gamma - 1)\|I - \alpha H\|\|\varphi - \varphi'\|_r + \|I - \alpha H\| \| \varphi - \varphi' \|_r \\ &\leq (2 + \gamma\|I - \alpha H\|)\|\varphi - \varphi'\|_r. \end{aligned}$$

Thus, F is a Lipschitz continuous function on $C([-r, 0], \mathbb{R}^{2n})$. There exists a single, continuous function $y(t)$ that serves as the solution to the equation system identified by (2.9).

$$\begin{aligned} &\|F(\varphi)\| = \|F(\varphi) - F(y^*)\| \\ &\leq 2\|\varphi - y^*\|_r + (\gamma - 1)\|I - \alpha H\|\|\varphi - y^*\|_r + \|I - \alpha H\|\|\varphi - y^*\|_r \\ &\leq (2 + \gamma\|I - \alpha H\|)\|\varphi - y^*\|_r \\ &\leq (2 + \gamma\|I - \alpha H\|)\|\varphi\|_r + (2 + \gamma\|I - \alpha H\|)\|y^*\|. \end{aligned}$$

Drawing upon the existence and extension theory of functional differential equations as outlined in [5], we demonstrate the uniqueness and continuity of a solution that indeed exists. Moreover, the solution $y(t)$ persists globally over the time interval $[0, +\infty)$. This conclusion completes our proof. □

Next, we proceed to delve into the global exponential stability characteristics of the solution pertaining to the DNN for SOCP problems.

For convenience, define $Z(t) = y(t) - y^*$ and

$$\begin{aligned} F(Z(t)) &= \Pi_C \left[y(t) - \alpha(Hy(t) + z)\right] - \Pi_C \left[y^* - \alpha(Hy^* + z)\right], \\ Z(t) &= \psi(t) := \varphi(t) - y^*, \quad t \in [-r, 0]. \end{aligned}$$

Then (2.9) can be rephrased as

$$\begin{cases} \frac{dZ(t)}{dt} = -\gamma Z(t) + (\gamma - 1)F(Z(t-r)) + F(Z(t)), \\ Z(t) = \psi(t), \quad t \in [-r, 0]. \end{cases} \tag{3.1}$$

We at first provide the following definition.

Definition 3.3. Assuming that the constants $M > 0$ and $a > 0$ fulfill the specified inequality:

$$\|y(t) - y^*\| \leq M\|\phi - y^*\|_r e^{-at}, \quad \forall t \geq 0,$$

The equilibrium point y^* of the DNN system, as defined in (2.9), exhibits global exponential stability, where

$$\|\psi - y^*\|_r = \sup_{-r \leq s \leq 0} \|\phi(s) - y^*\|.$$

An assumption is presented as follows:

Assumption 3.4. For $\alpha > 0$, $\gamma > 0$ and $r > 0$,

$$(1 + |\gamma - 1|e^{\gamma r})\|I + \alpha H\| - \gamma < 0$$

holds.

Under Assumption 3.4, we obtain the following global exponential stability result.

Theorem 3.5. *Under the condition that Assumption 3.4 is fulfilled, the equilibrium point of DNN system defined by (2.9) possesses global exponential stability.*

Proof. By (3.1), we have

$$\frac{dZ(t)}{dt} = -\gamma Z(t) + (\gamma - 1)F(Z(t-r)) + F(Z(t)),$$

which means that for $t \geq 0$

$$\int_0^t e^{\gamma s}\frac{dZ(s)}{ds}ds = \int_0^t -\gamma e^{\gamma s}Z(s)ds + \int_0^t (\gamma - 1)e^{\gamma s}F(Z(s-r))ds + \int_0^t e^{\gamma s}F(Z(s))ds. \tag{3.2}$$

Notice that

$$\int_0^t e^{\gamma s}\frac{dZ(s)}{ds}ds = e^{\gamma t}Z(t) - Z(0) - \int_0^t \gamma e^{\gamma s}Z(s)ds. \tag{3.3}$$

Combining (3.2) and (3.3), we have

$$Z(t) = e^{-\gamma t}Z(0) + e^{-\gamma t}\int_0^t (\gamma - 1)e^{\gamma s}F(Z(s-r))ds + e^{-\gamma t}\int_0^t e^{\gamma s}F(Z(s))ds.$$

Therefore, we obtain

$$\begin{aligned} \|Z(t)\| \leq e^{-\gamma t} \parallel \psi \parallel_r + \int_0^t |\gamma - 1|e^{\gamma(s-t)}\|F(Z(s-r))\|ds \\ + \int_0^t e^{\gamma(s-t)}\|F(Z(s))\|ds \end{aligned}$$

$$\leq e^{-\gamma t} \parallel \psi \parallel_r + |\gamma - 1| \int_0^t e^{\gamma(s-t)} \|I + \alpha H\| \|y(s-r) - y^*\| ds$$

$$+ \int_0^t e^{\gamma(s-t)} \|I + \alpha H\| \|y(s) - y^*\| ds$$

$$= e^{-\gamma t} \parallel \psi \parallel_r + (\gamma - 1) e^{\gamma r} \int_{-r}^{t-r} e^{\gamma(s-t)} \|I + \alpha H\| \|y(s) - y^*\| ds$$

$$+ \int_0^t e^{\gamma(s-t)} \|I + \alpha H\| \|y(s) - y^*\| ds$$

$$\leq e^{-\gamma t} \parallel \psi \parallel_r + |\gamma - 1| e^{\gamma r} \int_{-r}^{0} e^{\gamma(s-t)} \|I + \alpha H\| \|y(s) - y^*\| ds$$

$$+ (1 + |\gamma - 1| e^{\gamma r}) \int_0^t e^{\gamma(s-t)} \|I + \alpha H\| \|y(s) - y^*\| ds.$$

Consequently, we have

$$\begin{aligned} e^{\gamma t} \|y(t) - y^*\| \quad \leq \quad & \parallel \psi \parallel_r + \tfrac{|\gamma-1|}{\gamma} e^{\gamma r} (1 - e^{-\gamma r}) \|I \\ & + \alpha H\| \parallel \psi \parallel_r + \textstyle\int_0^t e^{\gamma s} (1 + |\gamma - 1| e^{\gamma r}) \|I + \alpha H\| \|y(s) - y^*\| ds. \end{aligned}$$

According to the Gronwall Theorem, we have

$$\|y(t) - y^*\| \leq M e^{\beta t},$$

where

$$M = \left(1 + \frac{|\gamma - 1|}{\gamma} e^{\gamma r} (1 - e^{-\gamma r}) \|I + \alpha H\|\right) \parallel \psi \parallel_r$$

and $\beta = (1 + |\gamma - 1| e^{\gamma r}) \|I + \alpha H\| - \gamma < 0$. □

Remark 3.6. Based on the aforementioned theorem, under the premise that Assumption 3.4 is met, the DNN model (2.9) introduced in this paper converges globally and exponentially to the sole optimal solution of the SOCP (2.1).

4. Numerical results and simulations

In this section, we will combine the theory of above section to conduct some specific numerical calculations, verify the effectiveness of the DNN for solving SOCP problems, and provide specific experimental data to visually demonstrate the stability of equilibrium point of the DNN for SOCP in a graphical manner. The numerical experiments in this part were conducted on a DELL laptop with an Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz and 2.90 GHz, using a 64 bit operating system and Matlab R2016a for simulations.

Let's first consider a simple example:

Example 1: Consider the following SOCP problem:

$$\begin{aligned} \min_{x} \quad & f(x) = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\ \text{s.t.} \quad & h(x) = Ax - b = 0, \quad g(x) = x \in K^6, \end{aligned}$$

where

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 4 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 9 \\ 20 \\ 6 \\ 4 \\ 8 \end{pmatrix}.$$

Obviously, this problem can be rewritten as a SOCP problem in the form of (2.1), and the optimal solution is $x^* = (3, 1, 2, 5, 3, 4)^T$. Next, we use the DNN to solve this problem. For $r = 0.1$, we use the `dde23` function in MATLAB to solve the original problem. Then use the `disp` function to output the solution, and use the `plot` function to draw a numerical solution graph, as shown in Figure 1, which illustrates the evolution of the numerical solution over time.

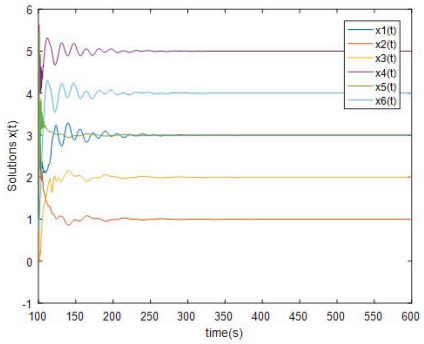

FIGURE 1. Evolution of $x(t)$ for Example 1 with $r = 0.1$

It can be observed very intuitively that the continuous solution of the equation exists and is unique, and the solution trajectory converges globally and exponentially, converging to $x^* = (3, 1, 2, 5, 3, 4)^T$. The stored data is specifically represented below in Table 1.

Next, we will further consider the influence of delay parameters r on the convergence speed of differential equation solutions. We take $r = 0.5, 1, 5$, in sequence and present the corresponding convergence results, as shown in Figures 2-4.

We can see from the graph that as r increases, the time required for the solution of the DNN to converge to the optimal solution becomes longer and longer. In fact, when $r = 5$ is present, a globally exponentially stable image can still be obtained, as shown in Figure 5.

But the computation time required to achieve this result has also significantly increased, taking nearly 10 seconds, far exceeding the computation time required to reach a stable solution for $r = 0.5$ and $r = 1$. That is to say, the delay in time in DNN directly affects the global exponential convergence speed of the DNN, and the larger the delay, the slower the convergence speed.

TABLE 1. The optimal solution x and the optimal value $f(x)$ of Example 1 corresponding to column i when $r = 0.1$

Step i	Optimal Solution x	Optimal Value $f(x)$
1	$(1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000)^T$	6.0000
50	$(3.0999, 2.7268, 0.0977, 4.6101, 3.2002, 1.4401)^T$	15.1748
200	$(3.2561, 0.8861, 2.1331, 4.8970, 2.9553, 3.9046)^T$	18.0322
500	$(2.9822, 1.0102, 1.9875, 5.0004, 3.0050, 3.9986)^T$	17.9839
750	$(2.9993, 1.0007, 1.9977, 4.9898, 3.0022, 3.9996)^T$	17.9893
1050	$(2.9998, 0.9998, 2.0000, 4.9997, 2.9994, 4.0000)^T$	17.9987
1066	$(3.0001, 1.0001, 2.0002, 5.0001, 3.0009, 3.9999)^T$	18.0013

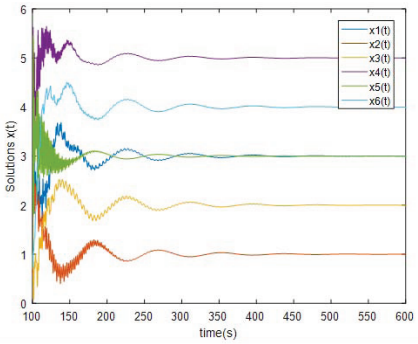

FIGURE 2. Evolution of $x(t)$ for Example 1 with $r = 0.5$

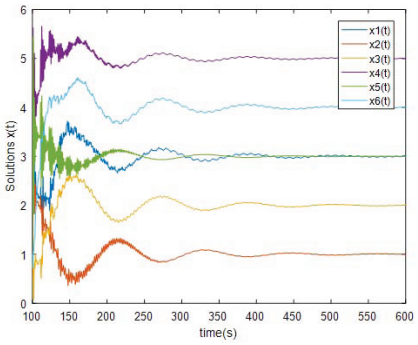

FIGURE 3. Evolution of $x(t)$ for Example 1 with $r = 1$

Next, we consider the deformation form of Example 1, changing the objective function from a linear function to a quadratic function, that is, solving the following problem.

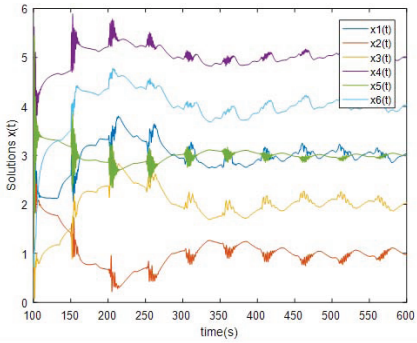

FIGURE 4. Evolution of $x(t)$ for Example 1 with $r = 5$

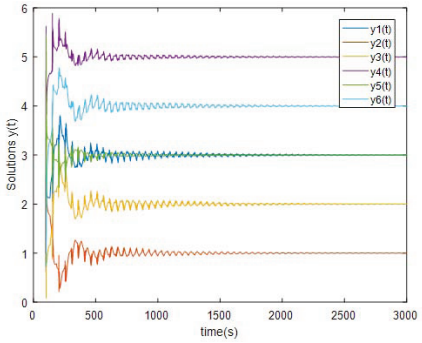

FIGURE 5. Evolution of $y(t)$ for Example 1 with $r = 5$

Example 2: Consider the following SOCP problem:

$$\begin{aligned} \min_{x} \quad & f(x) = \frac{1}{2}x_1^2 + x_2 + x_3 + x_4 + x_5 + x_6 \\ \text{s.t.} \quad & h(x) = Ax - b = 0, \quad g(x) = x \in K^6, \end{aligned}$$

where

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 4 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 9 \\ 20 \\ 6 \\ 4 \\ 8 \end{pmatrix}.$$

Considering the DNN with $\alpha = 0.5, \quad \gamma = 2, \quad \sigma = 1$. Then it can be calculated that

$$P = \begin{pmatrix} 0.9412 & 0.0588 & -0.1176 & -0.1765 & 0.0588 & -0.0588 \\ 0.0588 & 0.9412 & 0.1176 & 0.1765 & -0.0588 & 0.0588 \\ -0.1176 & 0.1176 & 0.7647 & -0.3529 & 0.1176 & -0.1176 \\ -0.1765 & 0.1765 & -0.3529 & 0.4706 & 0.1765 & -0.1765 \\ 0.0588 & -0.0588 & 0.1176 & 0.1765 & 0.9412 & 0.0588 \\ -0.0588 & 0.0588 & -0.1176 & -0.1765 & 0.0588 & 0.9412 \end{pmatrix},$$

$$Q = \begin{pmatrix} -0.0588 & -0.1765 & -1.0000 & 1.1765 & 0.8824 \\ 0.0588 & 0.1765 & 1.0000 & -0.1765 & -0.8824 \\ -0.1176 & -0.3529 & -0.0000 & 0.3529 & 0.7647 \\ -0.1765 & 0.4706 & 1.0000 & -0.4706 & -1.3529 \\ 0.0588 & 0.1765 & 0.0000 & -0.1765 & 0.1176 \\ 0.9412 & -0.1765 & -1.0000 & -0.8235 & 0.8824 \end{pmatrix},$$

$$z = (-1.4118, -2.5882, 1.1765, -0.2353, -4.5882, -2.4118)^T.$$

Let $r = 0.1$, use the `dde23` function in MATLAB again to solve and draw the trajectory graph of the numerical solution, as shown in Figure 6.

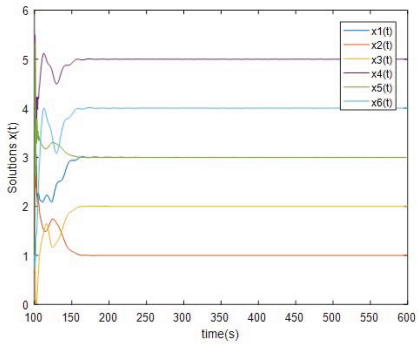

FIGURE 6. Evolution of $x(t)$ for Example 2 with $r = 0.1$

Similarly, it can be observed very intuitively that the continuous solution of the equation exists and is unique, and the equation is globally exponentially convergent, converging to x^*. Next, the stored data is specifically represented in Table 2.

TABLE 2. The optimal solution x and the optimal value $f(x)$ of Example 2 corresponding to column i when $r = 0.1$

Step i	Optimal Solution x	Optimal Value $f(x)$
1	$(1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000)^T$	6.5000
50	$(2.9738, 2.7955, 0.0399, 4.2190, 3.6235, 1.4258)^T$	19.4992
200	$(2.9336, 1.0622, 1.9283, 4.9655, 3.0270, 3.9362)^T$	22.1549
500	$(3.0001, 1.0001, 2.0001, 5.0003, 3.0003, 4.0001)^T$	22.5013
750	$(2.9996, 0.9997, 1.9998, 4.9993, 2.9992, 3.9997)^T$	22.4961
800	$(3.0004, 1.0003, 2.0003, 5.0007, 3.0001, 4.0000)^T$	22.5013
897	$(2.9999, 1.0000, 2.0000, 5.0000, 2.9999, 4.0001)^T$	22.4996

Due to the consistent constraints in Example 2 and Example 1, the optimal solution should also remain consistent. Next, we will further consider the effect of time delay on the convergence speed of DNN. Due to $0 \leq r < 2.9793$, and $r = 0.5, 1, 2$ is taken sequentially, the convergence results are shown in Figures 7-9.

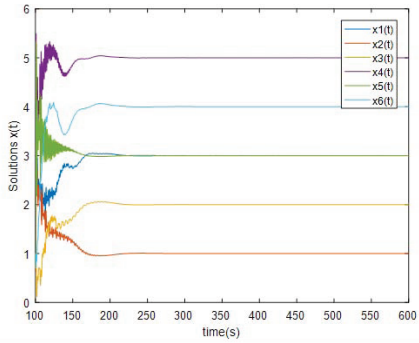

FIGURE 7. Evolution of $x(t)$ for Example 2 with $r = 0.5$

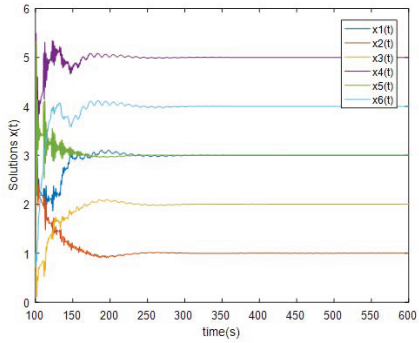

FIGURE 8. Evolution of $x(t)$ for Example 2 with $r = 1$

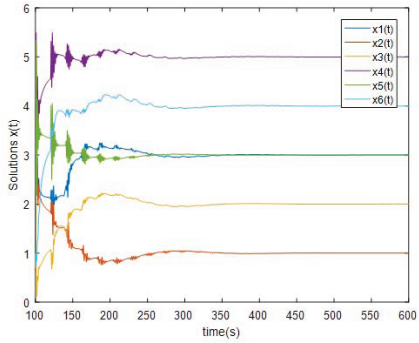

FIGURE 9. Evolution of $x(t)$ for Example 2 with $r = 2$

According to the graph, it can be intuitively seen that the delay in time in DNN directly affects the global exponential convergence speed of the DNN , and the larger the delay parameter r, the slower the convergence speed.

Finally, we consider a practical problem, namely the optimization problem of gripping force for multi finger robotic arms [18], which aims to find the minimum gripping force required for moving objects.

Example 3: Consider the following linear projection equation:

$$y = \Pi_C \left[y - \alpha(Qy + q) \right],$$

where

$$Q = \begin{bmatrix} 3 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{bmatrix}, \quad q = \begin{bmatrix} -2 \\ 0 \\ 0 \end{bmatrix}, \quad \alpha = 0.5.$$

In this example, we consider the following linear projection equation, whose exact solution is given by $x^* = [2,\ 1.2,\ 1.6]^\top$. We apply both a projection neural network

$$\frac{dy(t)}{dt} = \Pi_C \left[y(t) - \alpha(Qy(t) + q) \right] - y(t) \tag{4.1}$$

and the proposed delayed neural network (2.9) to solve this equation. The resulting solution trajectories are illustrated in Figure 10 and 11, respectively. As observed from the comparison, the DNN converges significantly faster and more stably than its non-delayed counterpart. While the projection neural network (4.1) suffers from oscillations and slow convergence due to the non-expansive nature of the projection mapping, the delayed model effectively suppresses such behavior and drives the state to the solution in a smoother and more efficient manner. This demonstrates the superiority of the DNN in handling projection-type equations.

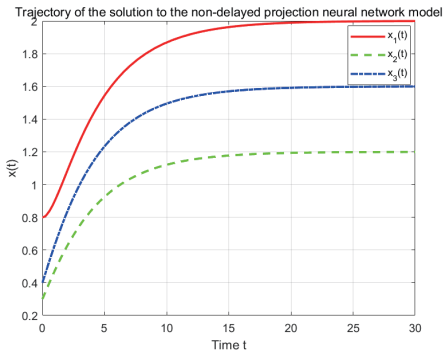

FIGURE 10. Evolution of $x(t)$ for Example 3 of non-DNN

Example 4: For a robotic arm with m fingers, the optimization problem of optimal grip force is as follows:

$$\begin{aligned} \min_{f} \quad & \frac{1}{2} f^T f \\ \text{s.t.} \quad & Gf = -f_{\text{ext}}, \\ & \|f_{i1}, f_{i2}\| \le \mu_{i3} f_{i3} \quad (i = 1, \ldots, m), \end{aligned}$$

where $f = [f_{11}, f_{12}, \cdots, f_{m3}]^T \in \mathbb{R}^{3m}$ represents the gripping force, $G \in \mathbb{R}^{6\times 3m}$ represents the grasping transformation matrix, f_{ext} represents the external wrench that changes with the movement time of the robotic arm, and μ_i represents the friction coefficient at position i.

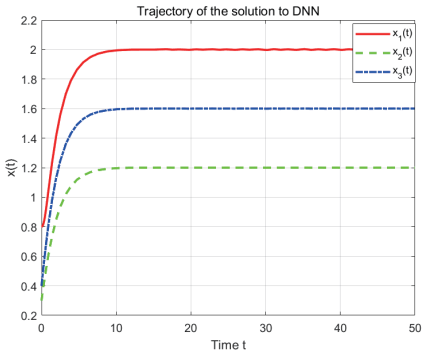

FIGURE 11. Evolution of $x(t)$ for Example 3 of DNN

As described in reference [18], we consider the gripping problem of a three finger robot, where the gripping positions of the fingers are $(0,1,1)^T$, $(1,0.5,0)^T$ and $(0,-1,0)^T$. For any i, we make the $\mu_i = \mu = 0.6$, robotic arm move along a vertical circular trajectory with a radius of r at a constant speed v.

To employ the DNN methodology, we reformulate the aforementioned problem into a SOCP problem:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T Q x \\ \text{s.t.} \quad & Ax = b, \quad x \in K^3 \times K^3 \times K^3, \end{aligned}$$

in which

$$Q = \operatorname{diag}\left(\frac{1}{\mu^2}, 1, 1, \frac{1}{\mu^2}, 1, 1, \frac{1}{\mu^2}, 1, 1\right),$$

and

$$A = \begin{pmatrix} 0 & 0 & 1 & -\frac{1}{\mu} & 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{\mu} & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & -0.5 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$b = \begin{pmatrix} 0 \\ -f_c \sin(\theta(t)) \\ Mg - f_c \cos(\theta(t)) \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

where $g = 9.8\,\mathrm{m/s^2}$, M is the mass of the object, $f_c = \frac{Mv^2}{L}$ represents centripetal force, t is the movement time of the robotic arm, $\theta = \frac{vt}{L} \in [0, 2\pi]$. Note that the problem is a nonlinear convex second-order cone programming problem, and the

matrix Q is positively definite. Let

$$M = 0.1\,\text{kg}, \quad L = 0.2\,\text{m}, \quad v = 0.4\pi\,\text{m/s},$$

When taking $t = 0\,s, 0.2\,s, 0.5\,s, 1\,s$, the convergence trajectory of the grasping force obtained from the DNN are shown in Figure 12, Figure 13, Figure 14, and Figure 15, respectively.

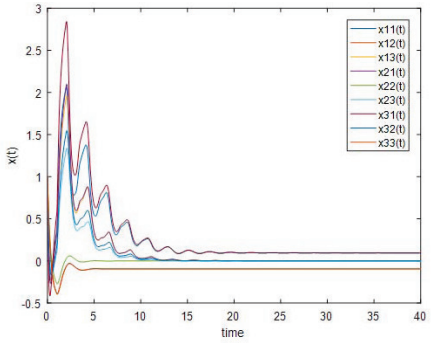

FIGURE 12. Evolution of $x(t)$ for Example 4 with $t = 0\,s$, $r = 1$

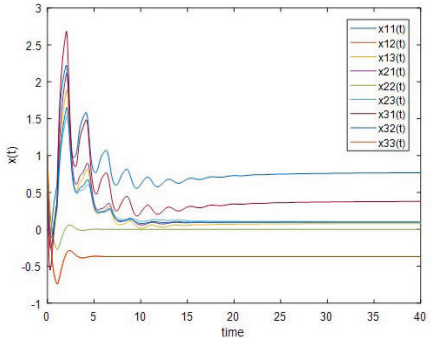

FIGURE 13. Evolution of $x(t)$ for Example 4 with $t = 0.2\,s$, $r = 1$

From the graph, we can obtain the optimal gripping force of the three finger machinery at different motion times. Due to the period of $\sin\theta(t)$ and $\cos\theta(t)$ being 2π, the solution trajectories of the DNN of $\theta = 0(t = 0s)$ and $\theta = 2\pi(t = 1s)$ are the same. We also find that the maximum gripping force occurs at the position of $\theta = \pi(t = 0.5s)$, corresponding to the maximum value of the downward wrench. The simulation experiment of the robotic arm grasping problem once again demonstrates that the DNN method is suitable for solving SOCP problems.

While conventional SOCP solvers can theoretically address static optimization problems at each fixed time instant, their reliance on repeated matrix factorizations and iterative operations makes them less suitable for real-time tasks involving continuously varying parameters. In dynamic environments, where optimization must be performed repeatedly and rapidly in response to changing inputs, such as in robotic control systems, traditional methods often struggle to meet real-time

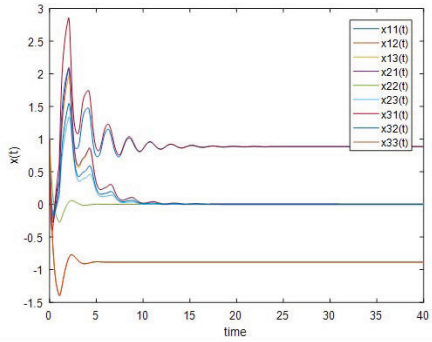

FIGURE 14. Evolution of $x(t)$ for Example 4 with $t = 0.5\,s$, $r = 1$

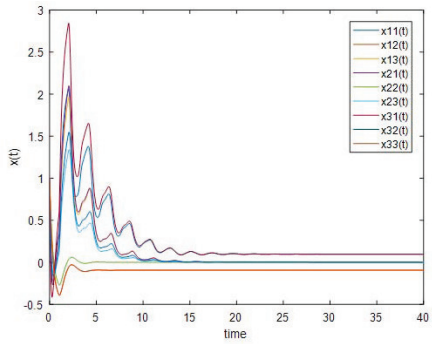

FIGURE 15. Evolution of $x(t)$ for Example 4 with $t = 1\,s$, $r = 1$

computational demands. In contrast, DNN model proposed in this paper offers a dynamic system-based framework that naturally adapts to parameter changes without the need for reinitialization or recomputation at each step. Its continuous-time dynamics can effectively track the optimal solution trajectory, while maintaining global exponential stability and strong robustness to time delays. Moreover, the architecture avoids Hessian matrix inversion and supports parallel, hardware-friendly implementation.

5. Conclusion

The present study introduces a novel category of DNNs designed specifically to tackle SOCP problems, establishing the existence and uniqueness of their solutions. Leveraging differential inequality methodologies, it has been rigorously demonstrated that the DNN provides a solution that exhibits global exponential convergence, thereby ensuring the reliability and stability of the proposed method. Finally, four simulation experiments were conducted to solve SOCP problems. Numerical calculations in MATLAB demonstrate that, und er the given assumptions, the conditions are met, the solution of the DNN exists uniquely and globally exponentially converges to the optimal solution of the SOCP problem. These verify the feasibility and effectiveness of this method in solving SOCP problems and highlight

its advantages in dealing with projection-type equations. It can be intuitively observed that the selection of delay parameters directly affects the convergence speed of the solution trajectory to the equilibrium point. The larger the delay parameter, the slower the convergence speed.

While the presented numerical simulations effectively demonstrate the theoretical soundness and computational feasibility of the proposed DNN model, further discussions on its practical deployment in real-world scenarios may enrich the paper's relevance. Beyond robotic grasping, the delayed neural network framework shows promise for broader real-time control systems where dynamic uncertainties, communication delays, and hardware constraints are present. Potential application domains include smart manufacturing lines, networked autonomous vehicles, aerial drone swarms, teleoperation platforms, and real-time energy management systems. The DNN's continuous-time structure, delay-robust stability, and hardware-parallelizable architecture make it a suitable candidate for embedded real-time optimization in complex engineering environments.

References

[1] F. Alizadeh and D. Goldfarb, *Second-order cone programming*, Math. Program. **95** (2003), 3–51.

[2] R. Andreani, E. H. Fukuda, G. Haeser, D. O. Santos and L. D. Secchin, *Optimality conditions for nonlinear second-order cone programming and symmetric cone programming*, J. Optim. Theory Appl. **200** (2024), 1–33.

[3] J. F. Bonnans and A. Shapiro, *Perturbation Analysis of Optimization Problems*, Springer, 2013.

[4] E. H. Fukuda, G. Haeser and L. M. Mito, *On the weak second-order optimality condition for nonlinear semidefinite and second-order cone programming*, Set-Valued Var. Anal. **31** (2023): 15.

[5] J. Hale, *Theory of Functional Differential Equations*, Springer-Verlag, New York, 1977.

[6] X. Hu, *Applications of the general projection neural network in solving extended linear-quadratic programming problems with linear constraints*, Neurocomputing **72** (2009), 1131–1137.

[7] F. Li, *Delayed Lagrangian neural networks for solving convex programming problems*, Neurocomputing **73** (2012), 2266–2273.

[8] L. Liang, D. F. Sun and K. C. Toh, *An inexact augmented Lagrangian method for second-order cone programming with applications*, SIAM J. Optim. **31** (2021), 1748–1773.

[9] Q. Liu, J. Cao and Y. Xia, *A delayed neural network for solving linear projection equations and its analysis*, IEEE Trans. Neural Netw. **16** (2005), 834–843.

[10] M. S. Lobo, L. Vandenberghe, S. Boyd and H. Lebret, *Applications of second-order cone programming*, Linear Algebra Appl. **284** (1998), 193–228.

[11] X. Miao, J. S. Chen and C. H. Ko, *A neural network based on the generalized FB function for nonlinear convex programs with second-order cone constraints*, Neurocomputing **203** (2016), 62–72.

[12] A. Nazemi, *A new collaborate neuro-dynamic framework for solving convex second order cone programming problems with an application in multi-fingered robotic hands*, Appl. Intell. **49** (2019), 3512–3523.

[13] A. Nazemi and A. Sabeghi, *A novel gradient-based neural network for solving convex second-order cone constrained variational inequality problems*, J. Comput. Appl. Math. **347** (2019), 343–356.

[14] A. Nazemi and A. Sabeghi, *A new neural network framework for solving convex second-order cone constrained variational inequality problems with an application in multi-finger robot hands*, J. Exp. Theor. Artif. Intell. **32** (2020), 181–203.

[15] J. Niu and D. Liu, *A new delayed projection neural network for solving quadratic programming problems subject to linear constraints*, Appl. Math. Comput. **219** (2012), 3139–3146.
[16] J. V. Outrata and D. Sun, *On the coderivative of the projection operator onto the second-order cone*, Set-Valued Anal. **16** (2008), 999–1014.
[17] C. Sha, H. Zhao and F. Ren, *A new delayed projection neural network for solving quadratic programming problems with equality and inequality constraints*, Neurocomputing **168** (2015), 1164–1172.
[18] J. Sun, W. Fu, J. H. Alcantara and J.-S. Chen, *A Neural Network Based on the Metric Projector for Solving SOCCVI Problem*, IEEE Trans. Neural Netw. Learn. Syst. **32** (2021), 2886–2900.
[19] X. Song, L. Zhou, Y. Wang, S. Zhang and Y. Zhang, *Stability analysis of proportional delayed projection neural network for quadratic programming problem*, Int. J. Math. **16** (2023): 2250070.
[20] X. Wen, *A Delayed neural network for solving a class of constrained pseudoconvex optimizations*, in] Proc. 9th Int. Conf. on Information Science and Technology (ICIST), IEEE, 2019, pp. 417–423.
[21] Y. Yang and J. Cao, *Solving quadratic programming problems by delayed projection neural network*, IEEE Trans. Neural Netw. **17** (2006), 1630–1634.
[22] Y. Zhang, *A projected-based neural network method for second-order cone programming*, Int. J. Mach. Learn. Cyber. **8** (2016), 1907–1914.
[23] L. Zhang and X. Sun, *Stability analysis of time-varying delay neural network for convex quadratic programming with equality constraints and inequality constraints*, arXiv:2107.06622, 2021.

Manuscript received January 5, 2025
revised June 4, 2025

Jie Zhang
School of Mathematics, Liaoning Normal University, Dalian 116029, China
E-mail address: jie_zhang@lnnu.edu.cn

Jiani Yang
School of Mathematics, Liaoning Normal University, Dalian 116029, China
E-mail address: jn_yanglnnu@163.com

Yifei Wang
School of Mathematics, Liaoning Normal University, Dalian 116029, China
E-mail address: yf_wanglnnu@163.com