Journal of Applied Analysis and Computation
Volume 16, Number 5, October 2026, 2392–2414
Website:http://www.jaac-online.com
DOI:10.11948/20250277

THE PROJECTION NEURAL NETWORK METHOD AND THE EULER METHOD FOR SOLVING THE SDLCP*

Jie Zhang[1,†], Wen Deng[1] and Shuxin Wang[1]

Abstract This work introduces an innovative neural network based on projection techniques to address the Semi-Definite Linear Complementarity Problem (SDLCP). The SDLCP arises frequently in fields such as optimization theory, economic modeling, engineering computations, and operational analysis. From a theoretical standpoint, it is proven that, under appropriate assumptions, the developed method guarantees both the existence and uniqueness of solutions, alongside ensuring asymptotic and exponential stability. The continuous-time model is discretized using the explicit Euler scheme, and its convergence properties are rigorously established. To validate the proposed projection-based neural network approach and the discretization scheme, various MATLAB numerical tests are conducted. The designed neural network presents a viable and effective strategy for tackling SDLCP, showcasing considerable promise for practical deployment across disciplines.

Keywords Semidefinite complementarity problem, projection neural network, Euler discretization.

MSC(2010) 90C33, 65K10, 68T07, 15A52, 34D20.

1. Introduction

The Complementarity Problem (CP) was first proposed by mathematicians Lemke and Howson in 1964 [11] during their research on Nash equilibria in game theory. Since then, it has rapidly developed into an important branch of optimization theory. Complementarity problems are widely encountered in various fields such as economics, engineering, operations research, variational inequalities, traffic equilibrium, and market equilibrium, and they possess significant theoretical value and practical application prospects. With the deepening of research, complementarity problems have been gradually extended to more complex and generalized frameworks, including the Nonlinear Complementarity Problem (NCP), Variational Inequalities (VI), Cone Complementarity Problem (CCP), and Semidefinite Complementarity Problem (SDCP).

This study addresses the SDLCP by developing a neural dynamic model based on projection operators, combined with Euler discretization for numerical implementation. Define S^n as the set of symmetric matrices of real numbers with the dimension $n \times n$, and let S^n_+ represent its positive semidefinite subset. For a linear operator $L : S^n \to S^n$ and a symmetric matrix $Q \in S^n$, the corresponding SDLCP seeks a matrix $X \in S^n$ satisfying the following semidefinite

†The corresponding author.
[1]School of Mathematics, Liaoning Normal University, 850 Huanghe Road, 116029 Dalian, China
*The work are supported by National Natural Science Foundation of China (12171219) and the Special Fund for Basic Scientific Research Expenses of Universities in Liaoning Province (LJ212410165006).
Email: jie_zhang@lnnu.edu.cn(J. Zhang), Wen_Denglnnu@163.com(W. Deng), shuxin_wang@163.com(S. Wang)

complementarity conditions

$$X \in S_+^n, \quad F(X) := L(X) + Q \in S_+^n, \quad \text{and} \quad \langle X, F(X)\rangle = 0, \tag{1.1}$$

where $\langle X, F(X)\rangle$ stands for the trace of $XF(X)$.

Over the past few years, neural-network-based approaches have found extensive use in addressing a broad range of complementarity problems, owing to their advantages such as inherent parallelism, dynamic evolution capabilities, and ease of hardware realization. Liao and Qi et al. [14] first proposed a neural network model based on unconstrained minimization reformulation for solving linear complementarity problems (LCP). By constructing an energy function with a specific structure and designing a first-order gradient descent neural dynamic system, they realized a dynamic approach to the solution of complementarity systems. Subsequently, Liao, Qi, and Qi extended this framework to the nonlinear complementarity problem (NCP) setting [13]. They introduced the Fischer–Burmeister function to construct a nonsmooth merit function, based on which they established a continuous-time neural network model. The Lyapunov stability and exponential convergence of the system were analyzed in depth. Although the dynamical system is mathematically nonsmooth, the squared error function was smoothly approximated to ensure the equivalence between convergence and the complementarity solution. Hou et al. [8] focused on a more general class of generalized vertical complementarity problems (GVCP), and proposed a neural network model based on log-exponential smoothing functions. The complementarity conditions, originally in max-type form, were reformulated into a smooth optimization problem, and the resulting dynamical system was proved to be asymptotically stable and consistent under certain conditions. Li et al. [12] investigated time-varying linear complementarity problems (TLCP) and introduced a family of Zhang-type neural dynamic models characterized by convergence within a finite time horizon. Considering real-world complementarity systems that evolve dynamically over time, they introduced three types of nonlinear activation functions and constructed robust neural network models capable of discretization and noise tolerance. Alcantara and Chen [2] introduced a novel type of neural network architecture to address nonlinear complementarity problems (NCPs), leveraging smoothed variants of the natural residual operator, such as approximations to the minimum function.

In summary, these studies demonstrate the adaptability and methodological diversity of neural networks in addressing various complementarity problems. As far as we are aware, neural network approaches have rarely been employed to address the SDLCP in existing literature.

Against this backdrop, this paper aims to solve the SDLCP using a projection neural network framework. A projection neural network described in continuous time is first formulated to convert the SDLCP into an evolving system, incorporating a projection mechanism to ensure that the solution trajectory is constrained to the space of symmetric positive semidefinite matrices. Specifically, spectral decomposition is employed to project matrices, thereby maintaining the feasibility of the solution throughout the network's evolution. To enhance its suitability for numerical computation, the proposed continuous-time dynamical formulation is discretized via the explicit Euler approach, yielding an iterative scheme based on a discretized projection model. Through modification of the integration step, scaling parameters, and other tuning coefficients, the proposed scheme effectively regulates system stability and ensures the trajectory progresses toward convergence.

The SDLCP has higher computational complexity in solution compared with NCP and CCP, mainly for two reasons: First, its decision variables are extended from finite-dimensional vectors (NCP/CCP) to $n \times n$ symmetric positive semidefinite matrices, and the feasible region S_+^n is

a complex non-polyhedral convex cone, making traditional iterative methods face the problem of matrix-level constraint satisfaction; second, its complementarity condition $\langle X, F(X)\rangle = 0$ is defined based on the trace of matrix product, and the inner product operation is more complex than the element-wise complementarity of NCP, increasing computational costs and the difficulty of convergence analysis. Traditional numerical methods for NCP/CCP cannot be directly extended to SDLCP, and dimension reduction processing will also lead to loss of computational efficiency. In contrast, the projection neural network method proposed in this paper directly constructs a dynamic system on the symmetric positive semidefinite matrix space and combines spectral decomposition to realize projection on S_+^n, which not only avoids dimension reduction errors but also effectively reduces the computational complexity of matrix-level operations through its inherent parallelism—this is the core advantage of this method for solving SDLCP compared with methods for NCP/CCP.

Projection neural network approaches provide notable strengths when applied to the SDLCP. Firstly, neural network models are typically formulated as differential equations, which are structurally simple and analytically tractable. This feature facilitates both theoretical analysis and model design. Secondly, common computational platforms such as MATLAB provide mature and efficient ODE solver toolboxes, which makes the proposed approach straightforward to implement and verify in practice. Last but not least, neural networks possess inherent parallelism and modularity, enabling convenient embedding into hardware systems for real-time applications and on-device computation. Overall, the projection neural network framework provides a promising and practical approach for solving SDLCP problems, combining theoretical soundness with engineering applicability.

This article is structured in the following manner.

Section 2 outlines the foundational mathematical concepts pertinent to the SDLCP. Section 3 introduces a newly designed projection-based neural model and examines its structural properties and asymptotic behavior in detail. Section 4 focuses on the theoretical validation of the proposed system, particularly concerning its dynamic stability. In Section 5, the continuous-time model is discretized via the explicit Euler scheme, leading to a discrete-time formulation along with a discussion of its convergence behavior. Section 6 presents numerical experiments that demonstrate the accuracy, robustness, and parameter sensitivity of both dynamic models. Finally, Section 7 summarizes the main findings, highlights existing limitations, and proposes several potential directions for future investigation.

2. Preliminaries

To thoroughly investigate the neural network-based methods for solving the SDLCP (1.1), and to carry out theoretical analysis on their stability and convergence, it is essential to first introduce the relevant mathematical tools and foundational concepts. This chapter reviews key theoretical elements including the basic properties of the space of symmetric positive semidefinite matrices, the definition and characteristics of projection operators, Lipschitz continuity, and strong monotonicity. These preliminaries provide the necessary theoretical foundation for constructing the neural network models and for conducting subsequent stability analysis and numerical implementation. Define a projection operator $P_{S_+^n} : S^n \to S_+^n$ as follows [18]

$$P_{S_+^n}(X) = \arg\min_{Y \in S_+^n} \|X - Y\|_F.$$

The norm is characterized as the Frobenius norm, that is $\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2} = \sqrt{\text{Trace}\,(A^\top A)}$. For a square matrix $A = [A_{ij}] \in R^{n\times n}$ or $C^{n\times n}$, the Trace of A, denoted by Trace(A), is computed by summing the entries along its main diagonal [7]

$$\text{Trace}(A) = \sum_{i=1}^{n} A_{ii}.$$

The inner product of two matrices A and B is defined as $\langle A, B\rangle$. Specifically, $\langle A, B\rangle = \text{Trace}(A^\top B)$, and this is equal to the sum $\sum_{i,j} A_{ij}B_{ij}$. We define the vectorization of a matrix $X \in S_+^n$ as the column-wise stacking of its entries into a vector in R^{n^2}, denoted by vec(X),

$$\text{vec}(X) = \begin{pmatrix} X_{11} \\ X_{21} \\ \vdots \\ X_{n1} \\ X_{12} \\ X_{22} \\ \vdots \\ X_{nn} \end{pmatrix} \in R^{n^2}.$$

We first introduce the following definitions, which characterize the Lipschitz continuity and strong monotonicity of the mapping functions, providing fundamental functional class conditions for the subsequent construction of the neural network.

Definition 2.1. [3] Suppose $F : S^n \to S_+^n$ is a mapping. The function $F(x)$ is considered to satisfy strong monotonicity when there exists a positive constant $\mu > 0$ such that, for any $X, Y \in S^n$, the inequality

$$\langle F(X) - F(Y), X - Y\rangle_F \geq \mu\|X - Y\|_F^2$$

is fulfilled. Here, μ is a strictly positive parameter, known as the strong monotonicity constant.

Based on the above definitions, we now introduce several fundamental lemmas related to the projection operator. These findings are essential for facilitating the design of the neural network and for analyzing its convergence behavior.

This lemma is the fundamental property of the projection operator on the positive semidefinite cone S_+^n, which is the theoretical basis for constructing the projection neural network in this paper. The first inequality reflects the monotonicity of the projection, and the second inequality proves that the projection operator is nonexpansive, this property is crucial for proving the Lipschitz continuity of the mapping $\Phi(X)$ and the uniqueness of the neural network solution in Section 4.

Lemma 2.1. [10] *For* $\forall X \in S^n, Y \in S_+^n$,

$$\left\langle X - P_{S_+^n}(X), P_{S_+^n}(X) - Y \right\rangle \geq 0,$$

and $\forall U, V \in S^n$,

$$\left\| P_{S_+^n}(U) - P_{S_+^n}(V) \right\|_F \leqslant \|U - V\|_F.$$

This lemma establishes the equivalence relationship between the projection operator and the SDLCP complementarity condition, which is the key to transforming the SDLCP into a dynamic projection equation in Section 3.

Lemma 2.2. [4] *If $S_+^n \subseteq S^n$, $X^* \in S_+^n$, $F : S^n \to S_+^n$ is a continuous mapping, then*

$$\langle X - X^*, F(X^*)\rangle \geqslant 0, \forall X \in S_+^n$$

holds if and only if

$$X^* = P_{S_+^n}\left(X^* - \alpha F(X^*)\right).$$

Proposition 2.1. *Assume $\tilde{F} : R^{n^2} \to R^{n^2}$ is a differentiable mapping, and suppose its Jacobian matrix $\mathcal{J}\tilde{F}(x)$ complies with the Lipschitz condition.Specifically, a constant $L > 0$ exists such that*

$$\left\|\mathcal{J}\tilde{F}(x_1) - \mathcal{J}\tilde{F}(x_2)\right\| \leq L\|x_1 - x_2\|, \quad \forall x_1, x_2 \in R^{n^2}.$$

Then, the $\|\mathcal{J}\tilde{F}(x)\|$ is bounded on any bounded closed set in R^{n^2}.

Proof. As $\mathcal{J}\tilde{F}(x)$ possesses Lipschitz continuity, it follows that it remains continuous on R^{n^2}. By the extreme value theorem, a continuous function whose domain consists of a bounded closed set (i.e., a compact set in R^{n^2}) must attain its extreme values. Therefore, the function $\|\mathcal{J}\tilde{F}(x)\|$ is bounded on any bounded closed set. That is to say, there is a constant $M > 0$ with the property that

$$\|\mathcal{J}\tilde{F}(x)\| \leq M$$

for all x in that bounded closed set. □

This lemma is used to simplify the calculation of the Jacobian matrix of the Lyapunov function in the exponential stability proof of Section 4. It shows that the integral function $g(x)$ constructed by the symmetric mapping $\tilde{F}(x)$ has a Jacobian matrix exactly equal to $\tilde{F}(x)$, which avoids the complex integral derivation in the stability analysis and is the key to proving the global exponential stability of the neural network (Theorem 4.4).

Lemma 2.3. *If $\tilde{F} : D \subset R^{n^2} \to R^{n^2}$ is differentiable on the open convex set D, define the function g:*

$$g(x) = \int_0^1 (x - x_0)^\top \tilde{F}\left(x_0 + t(x - x_0)\right) dt.$$

If $\tilde{F}(x)$ is symmetric, then $\mathcal{J}g(x) = \tilde{F}(x)$.

Proof. See Section 4 in [17]. □

After presenting the fundamental lemmas, we now provide two important propositions concerning the trajectory convergence and stability of the neural network, which offer theoretical guarantees for the subsequent neural network design.

Proposition 2.2. [22] *Let $\gamma^+(x_0) = \{x(t, x_0) \mid t \geq 0\}$ denote the trajectory of the neural network, and let K represent the set of its equilibrium points. If the trajectory $\gamma^+(x_0)$ is bounded, then the distance of $x(t)$ to the K satisfies*

$$\lim_{t\to\infty} \operatorname{dist}(x(t), K) = 0, \quad \text{where } \operatorname{dist}(x, K) := \inf_{y\in K} \|x - y\|.$$

Moreover, if the equilibrium set K contains a single point x^*, it follows that

$$\lim_{t\to\infty} x(t) = x^*.$$

In this case, if the system is Lyapunov stable, it is also globally asymptotically stable.

Proposition 2.3. [20] *Suppose that for every initial condition $x(t_0) \in \mathbb{R}^{n^2}$, the following inequality holds*

$$\|x(t) - x^*\| \le c_0\|x(t_0) - x^*\| \exp\left(-\eta(t - t_0)\right), \quad \forall t \ge t_0,$$

where $c_0 > 0$, $\eta > 0$. Then, the system is said to be globally exponentially stable.

3. Construction of the projection neural network

In the previous section, we introduced the fundamental definition of the SDLCP along with the necessary mathematical preliminaries. Based on these foundations, this section constructs a projection neural network model for solving the SDLCP. By incorporating a spectral projection operator, the original problem is transformed into a dynamic system, allowing the neural network to gradually approach the optimal solution through its evolution process. Furthermore, we analyze the structural properties of the proposed model and provide a theoretical foundation for its stability.

To understand the projection in detail, we begin by introducing the definition of spectral decomposition. Given a scalar $\alpha \in R$, let α^+ represent the non-negative projection of α, defined as $\alpha^+ := \max\{\alpha, 0\}$. This operation ensures that negative values of α are clipped to zero, maintaining only the non-negative part. This is a common operation in optimization and neural network training, where non-negative constraints are often necessary. Consider a matrix D whose diagonal entries are $d_1, d_2, \ldots, d_n$, i.e., $D = \mathrm{diag}(d_1, d_2, \ldots, d_n)$. For this instance, we define $D^+ := \mathrm{diag}(d_1^+, d_2^+, \cdots, d_n^+)$, where each diagonal element d_i is replaced by d_i^+, ensuring all diagonal elements are non-negative [19]. This operation can be seen as a form of rectification, forcing each diagonal element to remain non-negative.

Given $X \in S^n$, suppose it admits a spectral decomposition $X = TDT^\top$, where T is an orthogonal matrix and D is diagonal. Define $X^+ := TD^+T^\top$ as the result of applying spectral modification to the matrix X. Thus, the projection neural network for solving SDLCP can be described in the following way

$$\frac{dX}{dt} = P_{S_+^n}[X - \alpha F(X)] - X,$$

where α is a positive constant. The projection is handled using the aforementioned spectral decomposition, as shown below

$$P_{S_+^n}(X - \alpha F(X)) = (X - \alpha F(X))^+.$$

To facilitate the subsequent proof and computation of the stability, we introduce the following definitions

$$\begin{aligned}
&x = \mathrm{vec}(X), \quad x^* = \mathrm{vec}(X^*),\\
&\tilde{F}(x) = \mathrm{vec}(F(X)),\\
&\tilde{P}_{S_+^n}(x - \alpha\tilde{F}(x)) = \mathrm{vec}(P_{S_+^n}(X - \alpha F(X))).
\end{aligned}$$

With these definitions in place, the neural network is expressed as

$$\frac{dx}{dt} = \tilde{P}_{S_+^n}(x - \alpha F(x)) - x, \tag{3.1}$$

where $\alpha > 0$. In this way, we transform the continuous-time neural network model into a discretized form suitable for numerical solving, and further investigate its stability.

Building on this, we will next analyze the dynamic properties and explore its stability and convergence. Stability is a key factor in determining whether this method can effectively solve the SDLCP. Accordingly, we conduct a detailed investigation of the neural network's stability conditions and validate the theoretical findings through representative examples.

We will approach this from both theoretical and numerical perspectives, first providing the mathematical analysis and then validating its behavior under different parameter conditions through numerical experiments. By comparing various learning rates, initializations, and convergence speeds, we further demonstrate the application potential and practical performance of this neural network approach in solving SDLCP.

4. Stability analysis

To assess the practicality and robustness of the developed neural network framework for solving the SDLCP, it becomes essential to conduct a theoretical examination of the dynamic system's stability. The stability property is a fundamental factor in determining whether the neural model reliably converges to a valid solution. In this section, we establish criteria for both asymptotic and exponential stability of the proposed approach, utilizing Lyapunov-based analysis in conjunction with the spectral projection mechanism. The results obtained offer solid theoretical evidence supporting the convergence behavior of the constructed model.

Theorem 4.1. *For $\forall X_0 \in S^n$, if the mapping L is Lipschitz continuous, the differential equation has a unique solution.*

Proof. As noted by Hale [6], establishing the uniqueness of the neural network solution can be achieved by verifying that

$$\Phi(X) = P_{S_+^n}(X - \alpha F(X)) - X$$

is locally Lipschitz continuous. By Lemma 2.1, the projection mapping $P_{S_+^n}$ is nonexpansive,

$$\left\| P_{S_+^n}(X_1) - P_{S_+^n}(X_2) \right\|_F \leqslant \|X_1 - X_2\|_F, \forall X_1, X_2 \in S^n.$$

To investigate the Lipschitz property of the mapping Φ, let us consider two matrices $X_1, X_2 \in S^n$, and define

$$\tilde{X}_i := X_i - \alpha F(X_i), \quad i = 1, 2.$$

Then, we analyze the difference

$$\begin{aligned}
\|\Phi(X_1) - \Phi(X_2)\|_F &= \left\| P_{S_+^n}(\tilde{X}_1) - X_1 - \left(P_{S_+^n}(\tilde{X}_2) - X_2\right) \right\|_F \\
&= \left\| \left(P_{S_+^n}(\tilde{X}_1) - P_{S_+^n}(\tilde{X}_2)\right) - (X_1 - X_2) \right\|_F \\
&\leq \|P_{S_+^n}(\tilde{X}_1) - P_{S_+^n}(\tilde{X}_2)\|_F + \|X_1 - X_2\|_F \\
&\leq \|\tilde{X}_1 - \tilde{X}_2\|_F + \|X_1 - X_2\|_F.
\end{aligned}$$

Since the mapping L is Lipschitz continuous, the mapping F is Lipschitz continuous. To be specific, there exists $r > 0$ such that

$$\|F(X_1) - F(X_2)\|_F \leqslant r \|X_1 - X_2\|_F .$$

Hence,

$$\|\Phi(X_1) - \Phi(X_2)\|_F \leqslant (2 + \alpha r) \|X_1 - X_2\|_F ,$$

that is to say $\Phi(X)$ is Lipschitz on S_+^n. Thus $\Phi(X)$ is locally Lipschitz and the neural network has a single solution. □

Invoking from [3], we provide the proof of asymptotic stability for (3.1).

Theorem 4.2. *Suppose that $\tilde{F}(x)$ is a monotone, continuously differentiable mapping. Let $X_0 \in S_+^n$, and define $x_0 = \mathrm{vec}(X_0)$. Let $x(t, x_0)$ denote the trajectory of the neural network system (2) starting from the initial state x_0, with $x(t_0, x_0)$ being satisfied. Then, the positive orbit*

$$\gamma^+(x_0) := \{x(t, t_0) \mid t \geq 0\}$$

is bounded.

Proof. By Hale [6],we know from

$$\frac{dx}{dt} = \tilde{P}_{S_+^n}(x - \alpha F(x)) - x,$$

that

$$\frac{dx}{dt}e^t + xe^t = e^t \tilde{P}_{S_+^n}(x - \alpha F(x)),$$

which means

$$\frac{d}{dt}\left[x(t)e^t\right] = e^t \tilde{P}_{S_+^n}(x - \alpha F(x)).$$

By integrating the above equation, we have

$$\int_{t_0}^{t} \frac{d}{ds}\left[x(s)e^s\right] ds = \int_{t_0}^{t} e^s \tilde{P}_{S_+^n}(x - \alpha F(x)) ds,$$

which means

$$e^s x(s)|_{t_0}^{t} = \int_{t_0}^{t} e^s \tilde{P}_{S_+^n}(x - \alpha F(x)) ds.$$

Furthermore, we have

$$e^t x(t) - e^{t_0} x(t_0) = \int_{t_0}^{t} e^s \tilde{P}_{S_+^n}(x - \alpha F(x)) ds.$$

Therefore, let $t_0 = 0$,

$$x = e^{-t} x_0 + \int_0^t e^{s-t} \tilde{P}_{S_+^n}(x - \alpha F(x)) ds, \quad \forall t \geqslant 0.$$

Formulate the subsequent Lyapunov function,

$$V(x) = \alpha[\tilde{F}(x)]^T \left[x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right] - \frac{1}{2}\left\|x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right\|^2 + \frac{1}{2}\|x - x^*\|^2.$$

Clearly, $V(x^*) = 0$, in Lemma 2.1, let $X = x$, $Y = x - \alpha\tilde{F}(x)$, we have

$$\left[x - \alpha\tilde{F}(x) - \tilde{P}_{S_+^n}(x - \alpha F(x))\right]^\top \left[\tilde{P}_{S_+^n}(x - \alpha F(x)) - x\right] \geqslant 0,$$

which means

$$\left\langle -x + \alpha\tilde{F}(x) + \tilde{P}_{S_+^n}(x - \alpha F(x)), x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right\rangle \geqslant 0.$$

Split the first term,

$$\left\langle \alpha\tilde{F}(x), \quad x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right\rangle - \left\langle x - \tilde{P}_{S_+^n}(x - \alpha F(x)), x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right\rangle \geqslant 0,$$

by transposing terms and simplifying, we obtain

$$\alpha[\tilde{F}(x)]^\top \left[x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right] \geqslant \left\|x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right\|^2.$$

Therefore,

$$V(x) \geqslant \frac{1}{2}\left\|x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right\|^2 + \frac{1}{2}\|x - x^*\|^2$$

and

$$\|x - x^*\|^2 \leqslant 2V(x).$$

According to Thorem 3.2 in [5],

$$\mathcal{J}V(x) = \alpha\mathcal{J}\tilde{F}(x) + \left[\alpha\mathcal{J}\tilde{F}(x) - I_n\right]^\top \left[x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right] + (x - x^*).$$

Since $\tilde{F}(x)$ is monotonic, differentiable and continuous function, $\mathcal{J}V(x)$ is positive semidefinite. Therefore, along the solution orbit of the neural network (3.1), we obtain

$$\begin{aligned}
\frac{d}{dt}\left[V\left(x\left(t, x^0\right)\right)\right] &= [\mathcal{J}V(x)]^\top \frac{dx}{dt} \\
&= -\left[x - x^* + \alpha\tilde{F}(x)\right]^\top \left[x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right] \quad + \left\|x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right\|^2 \\
&\quad -\alpha\left[x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right]^\top \mathcal{J}F(x)\left[x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right] \qquad (4.1) \\
&\leq \left\|x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right\|^2 - \left[x - x^* + \alpha\tilde{F}(x)\right]^\top \left[x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right].
\end{aligned}$$

In Lemma 2.1, let $X = x^*$, $Y = x - \alpha\tilde{F}(x)$, we have

$$\left[x - \alpha\tilde{F}(x) - \tilde{P}_{S_+^n}[x - \alpha F(x))\right]^\top \left[\tilde{P}_{S_+^n}(x - \alpha F(x)) - x^*\right] \geqslant 0,$$

which means

$$\left[(x-x^*)+\alpha\tilde{F}(x)\right]^\top\left[x-\tilde{P}_{S_+^n}(x-\alpha F(x))\right]\geqslant\left\|x-\tilde{P}_{S_+^n}(x-\alpha F(x))\right\|^2+\alpha\,(x-x^*)^\top\tilde{F}(x). \quad (4.2)$$

$\forall X\in S_+^n$, that is $x\in R^{n^2}$, we have

$$\begin{aligned}(x-x^*)^\top\tilde{F}(x)&=(x-x^*)^\top\left[\tilde{F}(x)-F(x^*)\right]+(x-x^*)^\top\tilde{F}(x^*)\\&=(x-x^*)^\top\left[\tilde{F}(x)-\tilde{F}(x^*)\right]+x^\top\tilde{F}(x^*)\\&\geq(x-x^*)^\top\left[\tilde{F}(x)-\tilde{F}(x^*)\right].\end{aligned}$$

By (4.1) and (4.2), we have

$$\frac{d}{dt}\left[V\left(x\left(t,x^0\right)\right)\right]\leq-\alpha\,(x-x^*)^\top\left[\tilde{F}(x)-\tilde{F}\,(x^*)\right]\leq 0.$$

Therefore, $V\left(x\left(t,x^0\right)\right)$does not exhibit a monotonic increase with t, and we have

$$\|x\,(t,x_0)-x^*\|^2\leqslant 2V\,(x\,(t,x_0))\leqslant 2V\,(x_0)\,,$$

furthermore, we obtain

$$\|x\|\leq\|x^*\|+\sqrt{2V\,(x_0)}.$$

So, $\gamma^+\,(x_0)=\{x\,(t,t_0)\mid t\geqslant 0\}$ is bounded. □

Definition 4.1. [1] For a linear transformation $F(X)$, $F(X)$ has GUS-property if SDLCP has only one solution $\forall Q\in S^n$.

Theorem 4.3. *If $F(X)$ possesses the GUS-property, then $\langle x,\tilde{F}(x)\rangle=0$ admits a single solution x^*, which exhibits uniform asymptotic stability.*

Proof. Due to the fact that $F(X)$ has the GUS-property, this SDLCP has a unique solution X^*. From the above proof, it is already known that $V(x)$ is a function that approaches positive infinity, so, we have $\forall x(t)\neq x^*$,

$$\dot{V}(x(t))=\frac{d}{dt}[V(x(t))]\leqslant-\alpha\,(x-x^*)^\top\left[\tilde{F}(x)-\tilde{F}\,(x^*)\right]<0.$$

So the neural network (3.1) is uniformly asymptotically stables. □

Invoking from [21], we provide the proof of exponential stability for (1.1).

Theorem 4.4. *Suppose that the Jacobian of $\tilde{F}(x)$ is symmetric and satisfies a Lipschitz continuity condition. If $\tilde{F}(x)$ is strongly monotone over R^{n^2}, then the associated neural network exhibits exponential stability.*

Proof. Define the following Lyapunov function,

$$V_1(x)=\int_0^1(x-x^*)^\top\left[G\left(x^*+t\,(x-x^*)\right)-G\,(x^*)\right]dt,$$

where

$$G(x)=x+\alpha F(x).$$

Thus, we infer that

$$\frac{dV_1(x)}{dx} = x + \alpha\tilde{F}(x) - x^* - \alpha\tilde{F}(x^*).$$

By applying the mean value theorem, we get the following result

$$G(x + t(x - x^*)) - G(x^*) = \int_0^1 \mathcal{J}G(x + st(x - x^*))(x - x^*)\,\mathrm{sds}.$$

Therefore, we conclude that

$$V_1(x) = \int_0^1 \int_0^1 (x - x^*)^\top \mathcal{J}G(x + st(x - x^*))(x - x^*)\,\mathrm{sds}dt.$$

Since $\mathcal{J}\tilde{F}(x)$ satisfies the Lipschitz condition, thus $\mathcal{J}\tilde{F}(x)$ is bounded on R^{n^2}. Therefore, there exists $L > 0$ such that $\|\mathcal{J}\tilde{F}(x)\| \leq L$ holds. Then, we derive the relation

$$V_1(x) \leq (1 + L)\int_0^1 \int_0^1 \|x - x*\|^2 \mathrm{sds}dt \leqslant \frac{1 + L}{2}\|x - x^*\|^2.$$

According to (9) in [23], we have

$$\left[\tilde{P}_{S_+^n}(x - \alpha F(x)) - x^*\right]^\top F(x^*) \geqslant 0. \tag{4.3}$$

In Lemma 2.1, let $X = x - \alpha\tilde{F}(x), Y = x^*$, and we derive the relation

$$\left[\tilde{P}_{S_+^n}(x - \alpha F(x)) - x^*\right]^T \left[x - \alpha\tilde{F}(x) - \tilde{P}_{S_+^n}(x - \alpha F(x))\right] \geq 0. \tag{4.4}$$

It can be obtained from (4.3) and (4.4) that

$$\left[\tilde{P}_{S_+^n}(x - \alpha F(x)) - x^*\right]^\top \left[-\alpha\tilde{F}(x) + \alpha\tilde{F}(x^*) - \tilde{P}_{S_+^n}(x - \alpha F(x)) + x\right] \geqslant 0.$$

Then,

$$\begin{aligned}&\left[x - x^* + \alpha\tilde{F}(x) - \alpha\tilde{F}(x^*)\right]^\top \left[\tilde{P}_{S_+^n}(x - \alpha F(x)) - x\right]\\ &\leq -\alpha(x - x^*)^\top \left[\tilde{F}(x) - \tilde{F}(x^*)\right] - \left\|x - \tilde{P}_{S_+^n}(x - \alpha F(x))\right\|^2.\end{aligned}$$

Due to Definition 2.1 and Lemma 2.3, we obtion

$$\begin{aligned}\frac{dV_1(x)}{dt} &= \frac{dV_1(x)}{dx}\frac{dx}{dt}\\ &= \left[x - x^* + \alpha\tilde{F}(x) - \alpha\tilde{F}(x^*)\right]^\top \left[\tilde{P}_{S_+^n}(x - \alpha F(x)) - x\right]\\ &\leqslant -\alpha(x - x^*)^\top \left[\tilde{F}(x) - \tilde{F}(x^*)\right] - \left\|x - \tilde{P_{S_+^n}}(x - \alpha F(x))\right\|^2\\ &\leqslant -\alpha(x - x^*)^\top \left[\tilde{F}(x) - \tilde{F}(x^*)\right]\\ &\leqslant -k_1\|x - x^*\|^2\end{aligned}$$

$$\leqslant -kV_1(x),$$

where $k = \frac{2k_1}{1+L}$. Furthermore,

$$V_1(x) \leq V_1(x_0)\, e^{-k(t-t_0)}, \quad \forall t \geqslant t_0.$$

On the flip side,

$$\begin{aligned} V_1(x) &= \int_0^1 \int_0^1 (x-x^*)^\top \left(I_n + \alpha \mathcal{J}\tilde{F}\left(x + st\,(x-x^*)\right)\right)(x-x^*)\, s\, ds\, dt \\ &\geqslant \int_0^1 \int_0^1 \|x-x^*\|^2\, s\, ds\, dt \\ &= \frac{1}{2}\|x-x^*\|^2. \end{aligned}$$

Therefore, we obtion

$$\|x(t)-x^*\|^2 \leqslant \sqrt{2V_1(x_0)} e^{-k\frac{t-t_0}{2}}, \quad \forall t \geqslant t_0.$$

So the neural network (3.1) is exponentially stable. □

5. Euler method

In the context of numerical integration, the Euler discretization method is widely used to approximate solutions to differential equations. According to the definition in the paper Euler's Discretization Revisited by Maeda (1995) [16], Euler's method for discretizing a scalar ordinary differential equation of the form

$$\frac{dy}{dt} = g(y)$$

leads to the following iterative scheme

$$y_{k+1} = y_k + h\, g(y_k),$$

where y_k denotes the state of the system at the time step k, h is the chosen step size, and $g(y_k)$ defines the system's rate of change evaluated at y_k.

This explicit method is a first-order approximation based on the current value of the function. In our case, we apply Euler discretization to the differential equations governing the continuous-time neural network and incorporate the parameter δ. The resulting discretized equation is

$$x_{k+1} = x_k + \lambda_k \left[\tilde{P}_{S_+^n}\left(x_k - \alpha\delta F(x_k)\right) - x_k\right], \tag{5.1}$$

where x_k represents the system state at the k-th iteration, λ_k is the step size, α is a scaling parameter, δ is a tuning parameter, and $F(x_k)$ represents the continuous-time dynamics of the neural network.

Lemma 5.1. [15] *The discrete-time neural network described in equation (7) is said to exhibit global exponential convergence of rate η to the equilibrium point x^* if the sequence $\{x_k\}$ generated by the system satisfies*

$$\|x_k - x^*\| \leq \alpha \|x_0 - x^*\| e^{-\eta k}, \quad k = 1, 2, \cdots,$$

where $\eta > 0$ and $\alpha > 0$ are constants that do not depend on the initial condition x_0. Under this condition, system (5.1) *is globally exponentially stable.*

Invoking from [15], we provide the proof of the explicit Euler method.

Theorem 5.1. *The equation* (5.1) *is gobally exponentially convergent as long as the step sizethe step size* $0 < \lambda_k < 1/\alpha\delta M$ *for all* k. *In this context,* $\delta > 0$ *is a pre-defined constant that is small enough,* $M = \sup_{x \in S_+^n} \|\mathcal{J}F(x)\|$.

Proof. By the definition given in (5.1), it follows that

$$\begin{aligned}
\|x_{k+1} - x^*\| &= \left\| x_k + \lambda_k \left[\tilde{P}_{S_+^n} \left(x_k - \alpha\delta F\left(x_k\right) \right) - x_k \right] - x^* \right\| \\
&= \left\| x_k - x^* + \lambda_k \left[\tilde{P}_{S_+^n} \left(x_k - \alpha\delta F\left(x_k\right) \right) - x_k + x^* - x^* \right] \right\| \\
&= \left\| (1-\lambda_k)(x_k - x^*) + \lambda_k \left[\tilde{P}_{S_+^n} \left(x_k - \alpha\delta F\left(x_k\right) \right) - x^* \right] \right\| \\
&\leq (1-\lambda_k)\|x_k - x^*\| + \lambda_k \left\| \tilde{P}_{S_+^n} \left(x_k - \alpha\delta F\left(x_k\right) \right) - \tilde{P}_{S_+^n} \left(x^* - \alpha\delta F\left(x^*\right) \right) \right\| \\
&\leq (1-\lambda_k)\|x_k - x^*\| + \lambda_k \left\| x_k - \alpha\delta F\left(x_k\right) - x^* + \alpha\delta F\left(x^*\right) \right\| \\
&\leq (1-\lambda_k)\|x_k - x^*\| + \lambda_k \left\| x_k - x^* - \alpha\delta \mathcal{J}F(x_k)\left(x_k - x^*\right) \right\| \\
&= (1-\lambda_k)\|x_k - x^*\| + \alpha\lambda_k \left\| [I - \delta\mathcal{J}F(x_k)](x_k - x^*)) \right\| \\
&\leq (1 - \lambda_k\alpha\delta M)\|x_k - x^*\|.
\end{aligned}$$

For any iteration index k, suppose the step size satisfies $0 < \lambda_k < \frac{1}{\alpha\delta M}$. Then it follows that $0 < 1 - \lambda_k\alpha\delta M < 1$. Define the quantity $\eta_k := -\ln(1 - \lambda_k\alpha\delta M) > 0$, which allows us to write the following inequality

$$\|x_{k+1} - x^*\| \leq e^{-\eta_k}\|x_k - x^*\| \leq e^{-\eta_k(k+1)}\|x_0 - x^*\|.$$

This result shows that the discrete-time neural system described in equation (5.1) achieves global exponential convergence toward the equilibrium point x^*. □

The complete procedure for solving the SDLCP using the Euler discretization is outlined below.

Algorithm 5.1 Explicit Euler method for solving the SDLCP

Input: Set $k = 0$, step size sequence λ_k, parameters δ, α, maximum iterations K, tolerance ξ, initial matrix x_0.
Output: Approximate solution x^*.
Initialize $x_k = x_0$.
while $k < K$ **and** $\|x_{k+1} - x_k\|_F > \xi$ **do**
  Compute projection input:
  $\tilde{x} = x_k - \delta\alpha F(x_k)$
  Project onto the semi-definite cone:
  $P(x_k) = \text{SpectralProjection}(\tilde{x})$
  Update x_k using Explicit Euler:
  $x_{k+1} = x_k + \lambda_k(P(x_k) - x_k)$
  Update step size λ_k (adaptive strategy):
  if $\|x_{k+1} - x_k\|_F$ is increasing **then**
    $\lambda_k = \beta\lambda_k$ ▷ Reduce step size
  end if
  $k = k + 1$
end while
Return x_k as x^*.

This algorithm effectively enforces the positive semidefiniteness of the iterates via spectral projection at each iteration step. With appropriately chosen step sizes, scaling factors, and tuning parameters, the method demonstrates strong convergence and stability in numerical experiments. Furthermore, the incorporation of an adaptive step size strategy can enhance the overall computational efficiency.

6. Numerical experiment

In order to confirm the effectiveness and convergence property of the proposed projection neural network model (3.1) and the Euler method (5.1) in solving the SDLCP, this section evaluates the proposed approaches through numerical experiments. The influence of different parameter settings on the convergence speed, stability, and accuracy of the algorithms is analyzed. The experimental results show that the developed neural network models perform well numerically when dealing with SDLCP. The examples come from [1] and [9].

Example 6.1. Consider SDLCP(L, S_+^n, Q), where L is defined by

$$L(X) = AXA^\top$$

with

$$A = \begin{pmatrix} 17.25 & -1.75 & -1.75 & -1.75 & -1.75 \\ -1.75 & 16.25 & -2 & 0 & 0 \\ -1.75 & -2 & 16.25 & -2 & 0 \\ -1.75 & 0 & -2 & 16.25 & -2 \\ -1.75 & 0 & 0 & -2 & 16.25 \end{pmatrix},$$

$$Q = \begin{pmatrix} -9.25 & 1.25 & 1.25 & 1.25 & 1.25 \\ 1.25 & -8.25 & 1.5 & 0 & 0 \\ 1.25 & 1.5 & -8.25 & 1.5 & 0 \\ 1.25 & 0 & 1.5 & -8.25 & 1.5 \\ 1.25 & 0 & 0 & 1.5 & -8.25 \end{pmatrix}.$$

Take the initial point as

$$X_0 = \begin{pmatrix} 0.0620 & 0 & 0 & 0 & 0 \\ 0 & 0.0620 & 0 & 0 & 0 \\ 0 & 0 & 0.0620 & 0 & 0 \\ 0 & 0 & 0 & 0.0620 & 0 \\ 0 & 0 & 0 & 0 & 0.0620 \end{pmatrix}.$$

We implement the model in MATLAB and employ the ODE45 method to perform numerical integration of the projection neural network, thereby simulating the evolution of the solution trajectory.

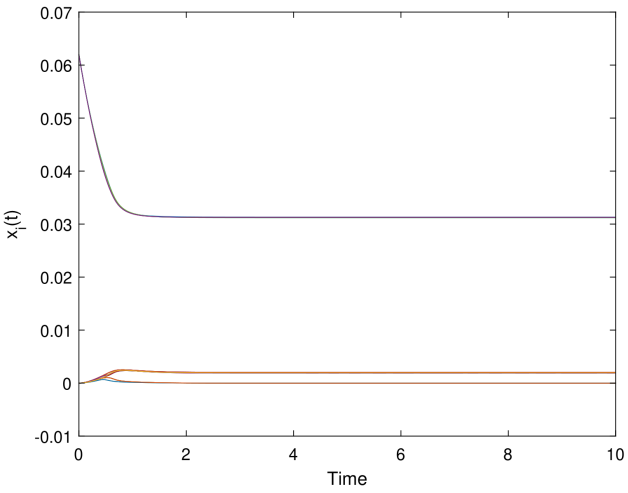

Figure 1. The evolution of $x_i(t)$ over time with $\alpha = 0.02$.

To further explore the influence of the step size parameter α on the network performance, we conduct comparative experiments with different α values.

Table 1. Impact of different α values on system behavior.

α **Value**	Convergence Speed	System Stability	Characteristic Behavior
0.005	Slow	Very smooth and stable	Slow convergence, minimal oscillation, smooth curve
0.02	Moderate	Stable	Faster convergence, off-diagonal components decay faster
0.1	Very fast	Very stable	Fastest convergence, rapid initial drop, no oscillation

The visualized trajectories corresponding to $\alpha = 0.0005$ and $\alpha = 0.1$ are shown in Figure 2 and Figure 3, which highlight the contrast in convergence speed and smoothness under different settings.

To verify the solution efficiency of the proposed method, a comparison is conducted with the primal-dual interior point method based on a new kernel function proposed by Abdessemed [1]on this example. The kernel function is as follows

$$g(x) = \frac{1}{2}\left(2x^2 + \frac{1}{x^2} - 5\right) + e^{\frac{1}{x}-1}.$$

As can be seen from the data in Table 5.3 of the literature [1], the fastest running time to obtain the solution is 6×10^{-2} s. In contrast, the average CPU time using the neural network (3.1) is 4×10^{-2} s, which is slightly faster than the time reported in the literature.

Beyond the continuous-time formulation, we further explore the behavior of the proposed neural network under a discrete-time implementation. The data derived from the iterative

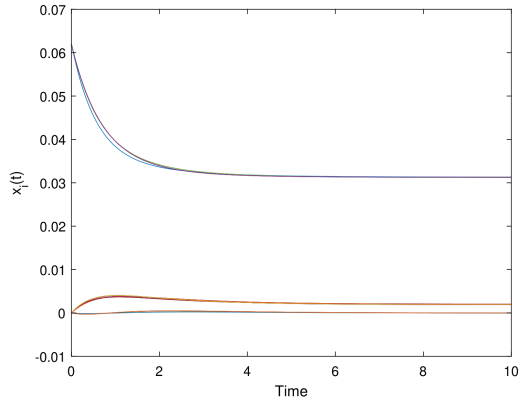

Figure 2. The evolution of $x_i(t)$ for $\alpha = 0.0005$.

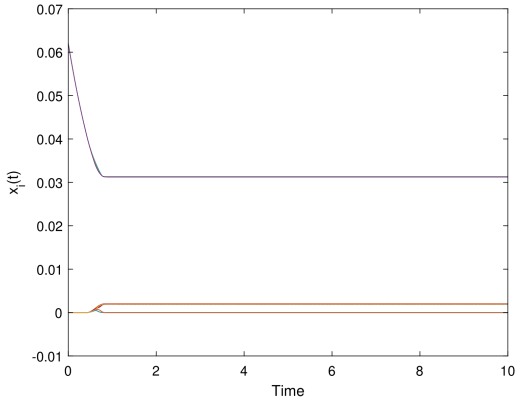

Figure 3. The evolution of $x_i(t)$ for $\alpha = 0.1$.

formula

$$x_{k+1} = x_k + \lambda_k \left[\tilde{P}_{S_+^n} \left(x_k - \alpha \delta F \left(x_k \right) \right) - x_k \right]$$

are presented as follows

$$X_1^* = \begin{pmatrix} 0.0314 & 0.0020 & 0.0021 & 0.0021 & 0.0020 \\ 0.0020 & 0.0314 & 0.0020 & 0.0000 & -0.0000 \\ 0.0021 & 0.0020 & 0.0313 & 0.0019 & 0.0000 \\ 0.0021 & 0.0000 & 0.0019 & 0.0313 & 0.0020 \\ 0.0020 & -0.0000 & 0.0000 & 0.0020 & 0.0314 \end{pmatrix}.$$

Figure 4 and Figure 5 show the Frobenius error and Heatmap visualization in the discrete-time case.

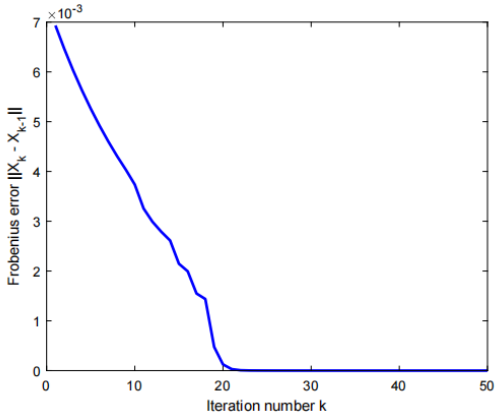

Figure 4. Frobenius error $||X_k - X_{k+1}||$ over iterations.

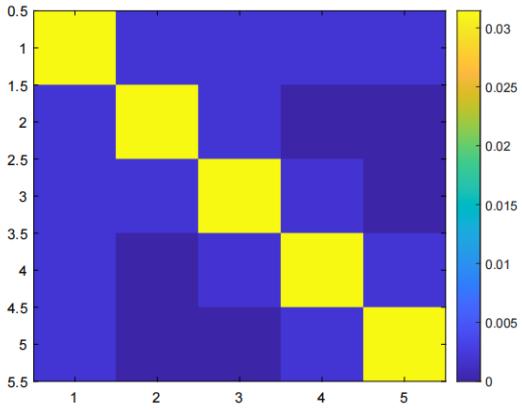

Figure 5. Heatmap visualization of the system.

To analyze how different parameters influence the discrete-time network behavior, we summarize their effects and suggested values in Table 2.

Table 2. Effect of parameters $(\lambda_k, \delta, \alpha)$ on system behavior.

Parameter	Suggested Range	Optimal Value	Impact on System
λ_k	$0.05 \to 0.005$	Decaying from 0.05 to 0.005	Speeds up early convergence and stabilizes later iterations
δ	$0.1 \sim 0.3$	$\delta = 0.2$	Larger δ accelerates updates but may cause oscillations
α	$0.3 \sim 0.5$	$\alpha = 0.5$	Controls the influence of $F(X)$ on projections

Figure 6 presents the Frobenius norm errors under various combinations of δ and α, offering further insights into the parameter sensitivity and convergence trends.

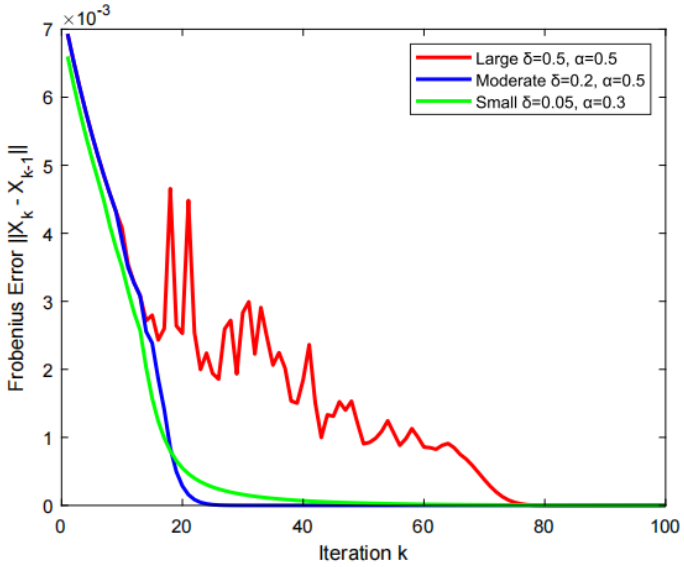

Figure 6. Frobenius error $||X_k - X_{k+1}||$ over iterations with varying δ and α.

Example 6.2. Consider SDLCP(L, S_+^n, Q), where L is defined by

$$L(X) = AX + XA$$

with

$$A = \begin{pmatrix} 3 & 1 \\ 1 & 1 \end{pmatrix}, \quad Q = \begin{pmatrix} -7 & -7 \\ -7 & 3 \end{pmatrix}.$$

Take the initial point as

$$X_0 = \begin{pmatrix} 0.5 & 0.5 \\ 0.3 & 1 \end{pmatrix}.$$

We implement the model in MATLAB to solve the problem. The optimal solution computed

in both cases is as follows.

$$X^* = \begin{pmatrix} 1.2017 & 0.6622 \\ 0.6622 & 0.3649 \end{pmatrix}.$$

The dynamic trajectory of the solution and the convergence of the discrete-time Frobenius error are visualized as follows.

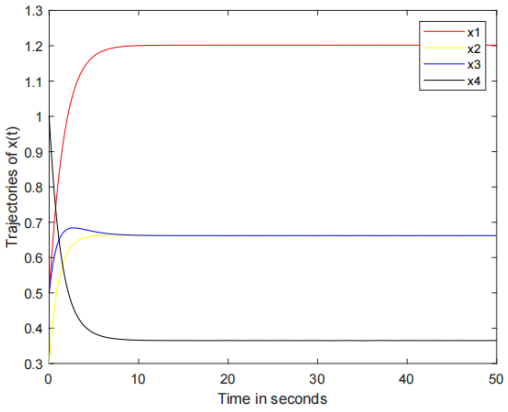

Figure 7. Trajectories of $x(t)$ over time.

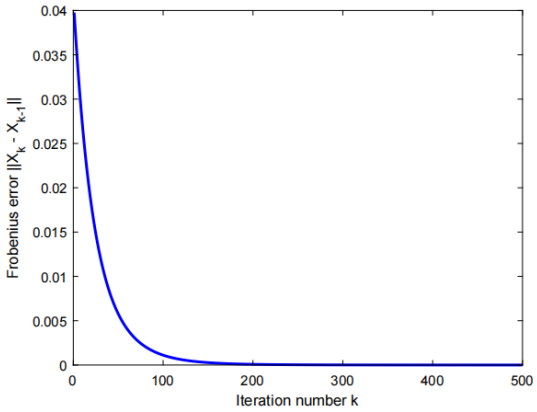

Figure 8. $||X_k - X_{k+1}||$ over iterations.

Example 6.3. Consider SDLCP(L, S_+^n, Q), where L and Q satisfy

$$L(X) = \frac{1}{2}\left(A^\top AX + XA^\top A\right)$$

and

$$Q = -\frac{1}{2}\left(A^\top C + C^\top A\right),$$

moreover,

$$A = \begin{pmatrix} 6 & -1 & 0 & 0 & 0 \\ -0.1 & 6 & -1 & 0 & 0 \\ 0 & -0.1 & 6 & -1 & 0 \\ 0 & 0 & -0.1 & 6 & -1 \\ 0 & 0 & 0 & -0.1 & 6 \\ 0 & 0 & 0 & 0 & -0.1 \end{pmatrix},$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -0.4 & 1 & 0 & 0 & 0 \\ -0.4 & -0.4 & 1 & 0 & 0 \\ -0.4 & 0 & -0.4 & 1 & 0 \\ -0.4 & 0 & 0 & -0.4 & 1 \\ -0.4 & 0 & 0 & 0 & 0.4 \end{pmatrix}.$$

Take the initial point as

$$X_0 = \begin{pmatrix} 0.2369 & 0 & 0 & 0 & 0 \\ 0 & 0.2369 & 0 & 0 & 0 \\ 0 & 0 & 0.2369 & 0 & 0 \\ 0 & 0 & 0 & 0.2369 & 0 \\ 0 & 0 & 0 & 0 & 0.2369 \end{pmatrix}.$$

We implement the model in MATLAB to solve the problem. The optimal solution computed in both cases is as follows.

$$X^* = \begin{pmatrix} 0.1639 & -0.0215 & -0.0342 & -0.0328 & -0.0300 \\ -0.0215 & 0.1553 & -0.0227 & -0.0019 & -0.0027 \\ -0.0342 & -0.0227 & 0.1558 & -0.0194 & 0.0013 \\ -0.0328 & -0.0019 & -0.0194 & 0.1563 & -0.0191 \\ -0.0300 & -0.0027 & 0.0013 & -0.0191 & 0.1576 \end{pmatrix}.$$

The dynamic trajectory of the solution and the convergence of the discrete-time Frobenius error are visualized in Figure 9 and Figure 10, respectively.

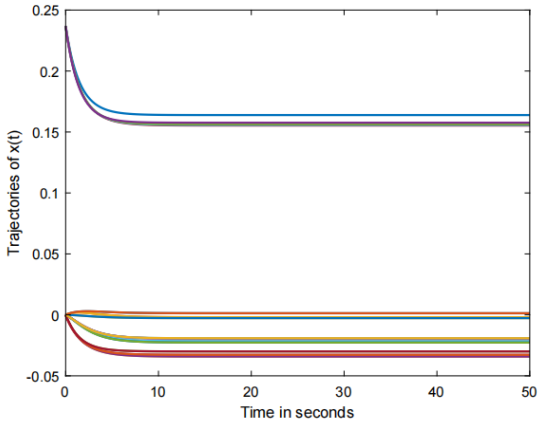

Figure 9. Trajectories of $x(t)$ over time.

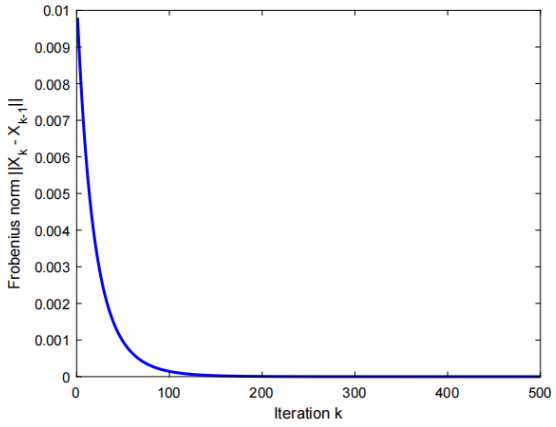

Figure 10. $||X_k - X_{k+1}||$ over iterations.

As shown in Table 5.5 of the literature [1], under the parameter settings of $\theta = 0.89$, $\alpha = 1$ and initial barrier parameter $\mu_0 = \mathrm{Tr}(ZR)/n$, the algorithm achieves convergence with 8 inner iterations and 7 outer iterations, and the corresponding running time is 4×10^{-2} s, which exhibits the optimal performance among all parameter combinations. In contrast, the average CPU time of the neural network-based method for this example is 2×10^{-2} s, which is faster than the fastest running time reported in Table 5.5 of [1].

Example 6.4. Let m be any positive integer, and $n = m^2$. Consider the SDLCP(L, S_+^n, Q), where

$$L(X) = AX + XA^T,$$

and

$$A = \hat{A} + \mu I_n \in R^{n\times n}, \quad \hat{A} = \text{Tridiag}(-1.5I_m, \hat{S}, -0.5I_m),$$
$$\hat{S} = \text{Tridiag}(-1.5, 4, -0.5) \in R^{m\times m}, \quad Q = -\mathbf{1}_{n\times n}.$$

Here, $\hat{A}$ is a tridiagonal matrix, and Q is a matrix of size $n \times n$ with all entries equal to -1.

The example is solved via the Euler discretization method (5.1), and its numerical error is analyzed with MATLAB. As shown in Figure 11, the error $\|X_{k+1} - X_k\|$ decreases monotonically

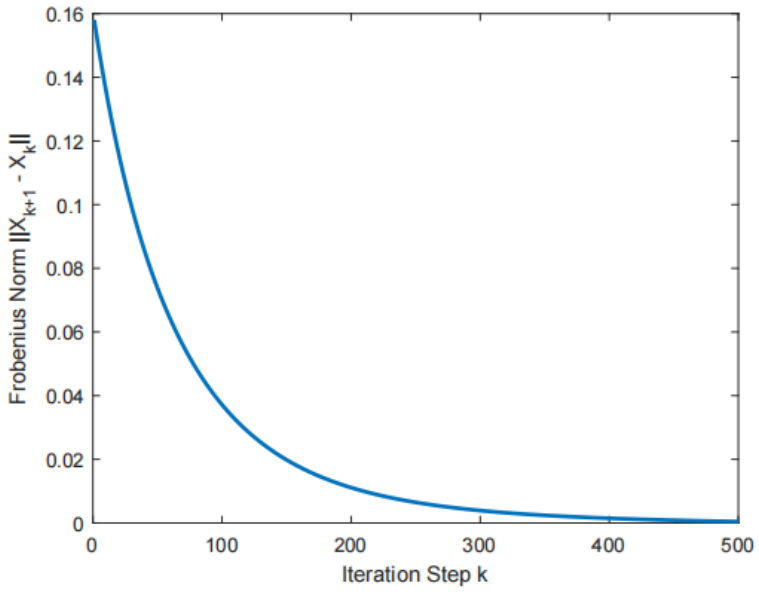

Figure 11. Frobenius error $||X_k - X_{k+1}||$ over iterations with varying δ and α.

with the number of iterations and eventually approaches zero, indicating that the proposed Euler method has good convergence and stability.

Example 6.5. Consider SDLCP(L, S_+^n, Q), where L is defined by

$$L(X) = AX + XA^\top$$

with

$$A = \begin{pmatrix} 1.7262 & -0.2347 & -0.3922 & -0.4775 & -0.4058 & -0.4664 \\ -0.1154 & 2.3244 & 0.4063 & -0.0747 & 0.0985 & -0.4312 \\ 0.0830 & 0.4827 & 2.3797 & -0.1873 & -0.0291 & -0.1804 \\ -0.2482 & 0.2302 & 0.3178 & 1.6615 & 0.1959 & 0.0309 \\ -0.2096 & -0.1561 & -0.2393 & -0.3212 & 2.1999 & 0.1544 \\ 0.1171 & 0.0841 & 0.0944 & -0.0771 & 0.1385 & 1.9076 \end{pmatrix},$$

$$Q = \begin{pmatrix} 0.1593 & -0.9954 & -1.2770 & -0.4402 & -0.7918 & -0.9241 \\ -0.9954 & 1.1645 & -1.6812 & -0.3341 & -1.1618 & -0.8573 \\ -1.2770 & -1.6812 & 0.6669 & -0.7393 & -1.1097 & -1.4273 \\ -0.4402 & -0.3341 & -0.7393 & 0.2363 & -0.8256 & -0.6725 \\ -0.7918 & -1.1618 & -1.1097 & -0.8256 & 0.2890 & -0.7654 \\ -0.9241 & -0.8573 & -1.4273 & -0.6725 & -0.7654 & 0.8210 \end{pmatrix}.$$

Take the initial point as

$$X_0 = \begin{pmatrix} 0.0620 & 0 & 0 & 0 & 0 \\ 0 & 0.0620 & 0 & 0 & 0 \\ 0 & 0 & 0.0620 & 0 & 0 \\ 0 & 0 & 0 & 0.0620 & 0 \\ 0 & 0 & 0 & 0 & 0.0620 \end{pmatrix}.$$

We implement the model in MATLAB and employ the ODE45 method to perform numerical integration of the projection neural network, thereby simulating the evolution of the solution trajectory. As illustrated in Figure 12, the trajectories tend toward steady values over time,

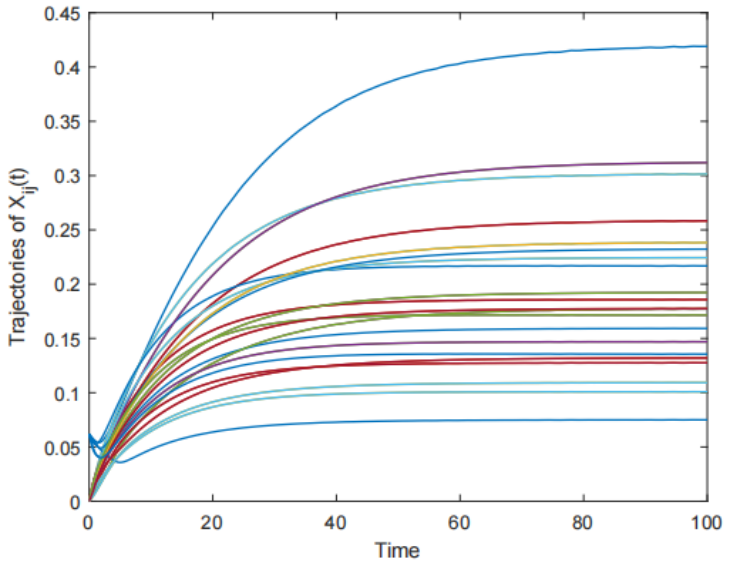

Figure 12. Trajectories of the components of $X(t)$.

demonstrating that the model (3.1) exhibits asymptotic stability and favorable convergence properties. This section validates both the effectiveness and stability of the proposed projection neural network and its Euler-discretized counterpart in solving the SDLCP, using three representative numerical experiments as evidence. These examples are designed from different dimensions and function structures. The experimental results demonstrate that both models

can accurately satisfy the complementarity and positive semidefiniteness constraints, successfully yielding the optimal solutions.

Specifically, the experimental results highlight the following points. The projection neural network exhibits good convergence behavior and smooth trajectory performance under different values of the parameter α, enabling stable approximation of the solution; The Euler-based discrete-time method not only inherits the convergence advantages of the continuous model, but also achieves higher computational efficiency with shorter runtime; Sensitivity analysis of parameters indicates that these parameters significantly affect the convergence speed and stability of the system. Proper tuning can balance convergence rate and oscillation magnitude.

Furthermore, the behavior of the algorithms is visualized and analyzed through Frobenius norm convergence plots, heatmap visualizations, and system trajectory evolution, which further confirm the theoretical results on stability. In conclusion, the framework for solving SDLCP demonstrates strong performance in terms of both accuracy and efficiency. Among the two, the discrete-time method generally shows faster convergence and higher numerical efficiency, indicating its practical value and application potential.

7. Conclusion

Despite the great potential of neural network methods in solving the SDLCP, several challenges remain in practical applications. For large-scale problems or those with complex constraints, improving the stability and convergence rate of neural networks remains a critical difficulty. In addition, the impact of parameter selection on algorithmic performance cannot be neglected.

Future research may be further explored in the following directions. Design more robust and adaptive network structures to enhance the capability of neural networks in handling high-dimensional and nonlinear SDCP. Enhance the theoretical examination of the stability and convergence characteristics of neural network algorithms, aiming to boost their reliability when deployed in practical scenarios. Explore the integration of modern optimization techniques such as reinforcement learning and meta-learning into the neural network framework to further enhance solving performance.

References

[1] N. Abdessemed, *Resolution of Semidefinite Linear Complementarity Problem (SDLCP) Using New Interior Point Methods Based on Kernel Functions*, University of Batna 2, Batna, 2023.

[2] J. H. Alcantara and J. S. Chen, *A new class of neural networks for NCPs using smooth perturbations of the natural residual function*, J. Comput. Appl. Math., 2022, 407, 114092.

[3] C. Dang, L. Qi and K. Yamamoto, *Neural networks for nonlinear and mixed complementarity problems and their applications*, Neural Netw., 2004, 17(2), 271–283.

[4] T. L. Friesz, K. Han, T. Y. L. Zhang and R. L. Tobin, *Day-to-day dynamic network disequilibria and idealized traveler information systems*, Oper. Res., 1994, 42(6), 1120–1136.

[5] M. Fukushima, *Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems*, Math. Program., 1992, 53, 99–110.

[6] J. K. Hale, *Ordinary Differential Equations*, Wiley-Interscience, New York, 1969.

[7] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 2012.

[8] B. Hou, J. Zhang and C. Qiu, *A neural network for a generalized vertical complementarity problem*, AIMS Mathematics, 2022, 7(4), 6650–6668.

[9] Y. F. Ke, *Convergence analysis on matrix splitting iteration algorithm for semidefinite linear complementarity problems*, Numer. Algorithms, 2021, 86(1), 257–279.

[10] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and their Applications*, SIAM, Philadelphia, 2000.

[11] C. E. Lemke and J. T. Howson, *Equilibrium points of bimatrix games*, J. Soc. Ind. Appl. Math., 1964, 12(2), 413–423.

[12] H. Li, S. Shao, S. Qin and Y. Yang, *Neural networks with finite-time convergence for solving time-varying linear complementarity problem*, Neurocomputing, 2021, 439, 146–158.

[13] L. Z. Liao, H. Qi and L. Qi, *Solving nonlinear complementarity problems with neural networks: A reformulation method approach*, J. Comput. Appl. Math., 2001, 131(1–2), 343–359.

[14] L. Z. Liao and H. D. Qi, *A neural network for the linear complementarity problem*, Math. Comput. Model., 1999, 29(3), 9–18.

[15] F. Liu, Y. Xia, X. Su and Z. Wang, *A discrete-time projection neural network for solving convex quadratic programming problems with hybrid constraints*, Int. J. Control Autom. Syst., 2023, 21(1), 328–337.

[16] Y. Maeda, *Euler's discretization revisited*, Proc. Japan Acad. Ser. A Math. Sci., 1995, 71(3), 58–61.

[17] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, SIAM, Philadelphia, 2000.

[18] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[19] P. Tseng, *Merit functions for semi-definite complementarity problems*, Math. Program., 1998, 83(1), 159–185.

[20] Y. Xia and G. Feng, *A new neural network for solving nonlinear projection equations*, Neural Netw., 2007, 20(5), 577–589.

[21] Y. Xia, H. Leung and J. Wang, *A projection neural network and its application to constrained optimization problems*, IEEE Trans. Circuits Syst. I, 2002, 49(4), 447–458.

[22] Y. Xia and J. Wang, *A recurrent neural network for solving linear projection equations*, Neural Netw., 2000, 13(3), 337–350.

[23] Y. Xia and J. Wang, *A general projection neural network for solving monotone variational inequalities and related optimization problems*, IEEE Trans. Neural Netw., 2004, 15(2), 318–328.

Received September 2025; Accepted March 2026; Available online April 2026.