Journal of Operational and Strategic Analytics
https://www.acadlore.com/journals/JOSA

Solving Single-Valued Trapezoidal Neutrosophic Linear Equations: An Embedding Approach for Uncertain Systems

Ahmed Abdelaziz Elsayed*, Arash Kermani Kolankeh

Department of Computer Engineering and Computational Sciences, School of Engineering, Applied Sciences and Technology, Canadian University Dubai, 117781 Dubai, United Arab Emirates

* Correspondence: Ahmed Abdelaziz Elsayed (ahmed.elsayed1516@gmail.com)

Received: 01-04-2025 **Revised:** 02-08-2025 **Accepted:** 02-15-2025

Citation: A. A. Elsayed and A. K. Kolankeh, "Solving single-valued trapezoidal neutrosophic linear equations: An embedding approach for uncertain systems," *J. Oper. Strateg Anal.*, vol. 3, no. 1, pp. 1–13, 2025. https://doi.org/10.56578/josa030101.

Abstract: Linear systems often involve coefficients that are uncertain or imprecise due to inherent variability and vagueness in the data. In scenarios where only approximate or vague knowledge of the system parameters is available, traditional fuzzy logic is commonly employed. However, conventional fuzzy logic may be inadequate when defining a membership degree with a single, precise value proves difficult. In such cases, Single-Valued Trapezoidal Neutrosophic Numbers (SVTrNNs) offer a more suitable framework, as they account for indeterminacy, alongside truth and falsity. The solution of Single-Valued Trapezoidal Neutrosophic Linear Equations (SVTrNLEs) was explored in this study using an embedding approach. The approach reformulates the SVTrNLEs into an equivalent crisp linear system, enabling the application of conventional solution methods. The solution was then obtained using either the matrix inversion method or the gradient descent optimization algorithm implemented in PyTorch. The robustness and adaptability of gradient-based optimization techniques were thoroughly assessed. The learning process minimizes the residual error iteratively, with convergence behaviour and numerical stability analyzed across various parameter configurations. The results demonstrate rapid convergence, proximity to exact solutions, and significant robustness to parameter variability, highlighting the efficacy of gradient descent for solving uncertain linear systems. These findings provide a foundation for the extension of gradient-based methods to more complex systems and broader applications. Furthermore, the existence and uniqueness of the neutrosophic solution to an $n \times n$ linear system were rigorously analyzed, with numerical examples provided to assess the reliability and efficiency of the proposed methods.

Keywords: Neutrosophic set; Neutrosophic trapezoidal number; Neutrosophic linear equation; Gradient descent

1 Introduction

Linear systems of equations are foundational in various scientific and engineering domains, providing solutions to problems ranging from structural analysis to machine learning. The system of linear equations is in the form of $AX = B$, where A is a matrix of coefficients, X is the vector of unknowns, and B is the vector of constants. This is fundamental in both pure and applied mathematics, with widespread practical applications [1]. Linear systems play a critical role across many disciplines. In engineering, they are used to model physical systems such as electrical circuits, structural mechanics, and fluid dynamics [2–5]. In economics, linear systems help model interrelationships in production, consumption, and equilibrium systems, particularly in input-output analysis [6]. In computer science, linear systems are crucial in optimization problems, machine learning, and scientific computing, where large-scale systems are solved to optimize performance and process complex datasets [7]. Efficient methods for solving linear systems are key to practical applications. Common techniques include Gaussian elimination, LU decomposition, iterative algorithms such as the Jacobi and Gauss-Seidel methods [8], and advanced numerical approaches like Krylov subspace methods and multigrid algorithms for solving large-scale problems in high-performance computing, where system size and complexity demand efficient computational methods [9]. The ability to solve these systems efficiently is essential for modern applications in scientific computing, data science, and artificial intelligence [10]. In practical scenarios, the coefficients of a linear system of equations may be uncertain or imprecise, often due to the inherent variability and vagueness in real-world data. When only vague or approximate knowledge about the actual values of

the parameters is available, it can be beneficial to use fuzzy logic to represent some or all of the system parameters. Fuzzy logic, introduced by Zadeh [11, 12] in his seminal works, extends traditional binary logic by incorporating degrees of membership. In fuzzy logic, the membership of an element to a set is represented by a real number between 0 and 1, reflecting the degree to which an element belongs to a set. This approach allows for a more flexible and nuanced representation of uncertainty compared to classical crisp values.

Following Zadeh's pioneering work, numerous researchers have explored the application of fuzzy logic in various fields, addressing the limitations of conventional approaches and improving the handling of imprecision and uncertainty in systems [13–15]. By integrating fuzzy logic into the analysis of linear systems, models that better accommodate and manage uncertainties in system parameters can be developed, leading to more robust and adaptive solutions. Several scholars have developed models for linear systems within a fuzzy environment, commonly referred to as fuzzy linear systems. The study of fuzzy linear systems dates back to at least 1980 [16]. Notably, Friedman et al. [17] proposed a specific model for addressing fuzzy linear systems, characterized by a crisp coefficient matrix and an arbitrary fuzzy number vector on the right-hand side. This model has since been modified and expanded upon by various researchers, as detailed in several studies [18–30] and their associated references.

Zadeh's fuzzy sets (FSs), while useful, are limited in cases where it is difficult to define the membership degree with a single specific value. To address the challenge of undefined non-membership degrees, Atanassov [31] introduced an extension known as Intuitionistic Fuzzy Sets (IFSs). Although IFSs can handle incomplete information in various real-world applications, they are not capable of addressing all types of uncertainty, particularly indeterminate and inconsistent information. To overcome these limitations, Smarandache [32] proposed the Neutrosophic Set (NS), a comprehensive framework that generalizes classical sets, FSs, interval-valued FSs, IFSs, and interval-valued IFSs. NSs are equipped to handle uncertain, indeterminate, and inconsistent information by explicitly quantifying indeterminacy. In NS theory, the truth membership, indeterminacy membership, and falsity membership are independent of one another. This allows NSs to effectively describe uncertain, incomplete, and inconsistent information, surpassing the limitations of existing methods in capturing uncertain decision data. The neutrosophic concept is further divided into two categories: Neutrosophic Numbers (NNs) and NSs.

While several methods have been developed to address problems using NSs, and some models have been proposed to solve linear systems involving NNs [33–37], to the best of our knowledge, systems of SVTrNNs have not yet been studied using Trapezoidal Neutrosophic Numbers (TrNNs). Therefore, a model for solving systems SVTrNNs was introduced in this study. The proposed method is based on the (α, β, γ)-cut approach and has a straightforward structure. The solution was obtained using the matrix inverse method and the gradient descent algorithm. The learning process minimizes the residual error iteratively, with convergence behavior and numerical stability analyzed across multiple parameter configurations. The results show rapid convergence, small distances from exact solutions, and robustness to parameter variability, demonstrating the effectiveness of gradient descent for such problems. These findings lay the groundwork for extending gradient-based methods to more complex systems and applications.

This study is organized as follows: Section 2 introduces fundamental concepts, definitions, and arithmetic operations related to TrNNs. Section 3 presents the SVTrNLEs and proposes a general model for a solution. Section 4 provides numerical experiments to demonstrate the method's reliability and efficiency. Sections 5 and 6 provide the results and discussion. Lastly, conclusions are drawn in Section 7.

2 Some Basic Definitions and Arithmetic Operations

Section 2 discusses some basic definitions related to NSs and single-valued NNs, respectively [32].

Definition 1. NS: Let X be a space of points (objects), with a generic element in X denoted by x [32]. A NS A in X is characterized by a truth-membership function $T_A(x)$, an indeterminacy-membership function $I_A(x)$, and a falsity-membership function $F_A(x)$. The functions $T_A(x)$, $I_A(x)$, and $F_A(x)$ are real standard or non-standard subsets of $]0^-, 1^+[$. That is $T_A(x)$:

$X \to]0^-, 1^+[, I_A(x) : X \to]0^-, 1^+[$, and $F_A(x) : X \to]0^-, 1^+[$. There is no restriction on the sum of $T_A(x)$, $I_A(x)$, and $F_A(x)$, therefore:

$$0^- \leq \sup T_A(x) + \sup I_A(x) + \sup F_A(x) \leq 3^+$$

Definition 2. A single-valued neutrosophic set (SVNS) is an instance of a NS [38]. Let X be a space of points (objects), with a generic element in X denoted by x. If the functions $T_A(x)$, $I_A(x)$ and $F_A(x)$ in Definition 1 are singleton subintervals/subsets in the real standard [0, 1], that is $T_A(x)$: $X \to [0, 1], IA(x) : X \to [0, 1]$, and $FA(x) : X \to [0, 1]$. Then, a SVNS A is denoted by:

$$A = \{(x, T_A(x), I_A(x), F_A(x)) \mid x \in X\}$$

This satisfies the condition of $0 \leq T_A(x) + I_A(x) + F_A(x) \leq 3$.

Definition 3. A SVTrNN is defined as $A^{\aleph} =< (a,b,c,d),(\mu,v,\omega) >$, whose truth membership function $T_{A^{\aleph}}(x)$, indeterminacy-membership function $I_{A^{\aleph}}(x)$, and falsity-membership function $F_{A^{\aleph}}(x)$ are given as follows [39]:

$$T_{A^{\aleph}}(x) = \begin{cases} \frac{(x-a)}{(b-a)}\mu & a \le x < b, \\ \mu & b \le x < c \\ \frac{(d-x)}{(d-c)}\mu & c \le x < d \\ 0 & otherwise. \end{cases}$$

$$I_{A^{\aleph}}(x) = \begin{cases} \frac{(b-x)+(x-a)v}{(b-a)} & a \le x < b, \\ v & b \le x < c \\ \frac{(x-c)+(d-x)v}{(d-c)} & c \le x < d, \\ 1 & otherwise. \end{cases}$$

$$F_{A^{\aleph}}(x) = \begin{cases} \frac{(b-x)+(x-a)\omega}{(b-a)} & a \le x < b, \\ \omega & b \le x < c, \\ \frac{(x-c)+(d-x)\omega}{(c-b)} & c \le x < d, \\ 1 & otherwise. \end{cases}$$

where, μ, v, ω represent the degree of truth, the degree of indeterminacy and the degree of falsity, respectively, and $0 \le \mu, v, \omega \le 1, 0 \le T_{A^{\aleph}}(x) + I_{A^{\aleph}}(x) + F_{A^{\aleph}}(x) \le 3, x \in A^{\aleph}$.

Definition 4. Let $A_1^{\aleph} =< (a_1,b_1,c_1,d_1),(\mu_1,v_1,\omega_1) >$ and $A_2^{\aleph} =< (a_2,b_2,c_2,d_2),(\mu_2,v_2,\omega_2) >$ be two SVTrNNs [39]. Then the arithmetic relations are defined as:

$$(a) A_1^{\aleph} \oplus A_2^{\aleph} =< (a_1+a_2, b_1+b_2, c_1+c_2, d_1+d_2), (\mu_1+\mu_2-\mu_1\mu_2, v_1v_2, \omega_1\omega_2) > \tag{1}$$

$$(b) \lambda A_1^{\aleph} = \begin{cases} < (\lambda a_1, \lambda b_1, \lambda c_1, \lambda d_1), (\mu_1, v_1, \omega_1) >, \text{ if } \lambda > 0 \\ < (\lambda d_1, \lambda c_1, \lambda b_1, \lambda a_1), (\mu_1, v_1, \omega_1) >, \text{ if } \lambda < 0 \end{cases} \tag{2}$$

Definition 5. If $A^{\aleph} =< (a,b,c,d),(\mu,v,\omega) >$, then (α,β,γ)-cut is given by [39]:

$$A^{\aleph}_{(\alpha,\beta,\gamma)} = \left\langle \begin{array}{c} [(a+\alpha(b-a))\mu, (d-\alpha(d-c))\mu], \\ [(b-\beta(b-a))v, (d+\beta(d-c))v], \\ [(b-\gamma(b-a))\omega, (d+\gamma(d-c))\omega] \end{array} \right\rangle \tag{3}$$

3 System of SVTrNLEs

In this section, two different methods for solving the SVTrNLEs are discussed. After converting SVTrNLEs into an equivalent system of equations, the solution was obtained directly using an embedding approach and the matrix inversion and this solution was approximated numerically using gradient descent in PyTorch.

Definition 6. The matrix equation is in the following form:

$$Ax^{\aleph} = B^{\aleph} \tag{4}$$

where, the coefficient matrix $A = (a_{ij})$ is a crisp $n \times n$ matrix and $b_i^{\aleph}, i = 1, 2, \ldots, n$ is a NN that is called a neutrosophic linear equation (NLE).

Definition 7. The system of equations is in the following form:

$$\begin{cases} a_{11}x_1^{\aleph} + a_{12}x_2^{\aleph} + \ldots\ldots\ldots\ldots\ldots + a_{1n}x_n^{\aleph} = b_1^{\aleph}, \\ a_{21}x_1^{\aleph} + a_{22}x_2^{\aleph} + \ldots\ldots\ldots\ldots\ldots + a_{2n}x_n^{\aleph} = b_2^{\aleph}, \\ . \\ . \\ . \\ a_{n1}x_1^{\aleph} + a_{n2}x_2^{\aleph} + \ldots\ldots\ldots\ldots\ldots + a_{nn}x_n^{\aleph} = b_n^{\aleph}. \end{cases} \tag{5}$$

This is called a system of neutrosophic linear equations (SNLE).

In the following Section 3.1, an embedding approach for solving the SNLE was proposed.

3.1 An embedding approach for solving SNLE

Let the solution of the SNLE of Eq. (5) be $x^{\aleph}$ and its (α,β,γ)-cut be $x^{\aleph}_{(\alpha,\beta,\gamma)} = \left(\left[\underline{x}^T(\alpha),\bar{x}^T(\alpha)\right],\ \left[\underline{x}^I(\beta),\bar{x}^I(\beta)\right],\left[\underline{x}^F(\gamma),\bar{x}^F(\gamma)\right]\right)$. If the (α,β,γ)-cut of b^{K} be $x^{\aleph}_{(\alpha,\beta,\gamma)} = \left(\left[\underline{b}^T(\alpha),\bar{b}^T(\alpha)\right],\left[\underline{b}^I(\beta),\bar{b}^I(\beta)\right],\left[\underline{b}^F(\gamma),\bar{b}^F(\gamma)\right]\right)$, then the SNLE of Eq. (5) can be written as:

$$\begin{cases} \sum_{j=1}^{n} a_{ij}x_j^T(\alpha) = \sum_{j=1}^{n} \underline{a_{ij}x_j^T(\alpha)} = \underline{b_i^T(\alpha)} \\ \overline{\sum_{j=1}^{n} a_{ij}x_j^T(\alpha)} = \sum_{j=1}^{n} \overline{a_{ij}x_j^T(\alpha)} = \overline{b_i^T(\alpha)} \\ \sum_{j=1}^{n} a_{ij}x_j^I(\beta) = \sum_{j=1}^{n} \underline{a_{ij}x_j^I(\beta)} = \underline{b_i^I(\beta)} \\ \overline{\sum_{j=1}^{n} a_{ij}x_j^I(\beta)} = \sum_{j=1}^{n} \overline{a_{ij}x_j^I(\beta)} = \overline{b_i^I(\beta)} \\ \sum_{j=1}^{n} a_{ij}x_j^F(\gamma) = \sum_{j=1}^{n} \underline{a_{ij}x_j^F(\gamma)} = \underline{b_i^F(\gamma)} \\ \overline{\sum_{j=1}^{n} a_{ij}x_j^F(\gamma)} = \sum_{j=1}^{n} \overline{a_{ij}x_j^F(\gamma)} = \overline{b_i^F(\gamma)} \end{cases} \tag{6}$$

If $x_i^{\aleph} = \left(\underline{x}_1^T,\ldots,\underline{x}_n^T,\bar{x}_1^T,\ldots,\bar{x}_n^T,\underline{x}_1^I,\ldots,\underline{x}_n^I,\bar{x}_1^I,\ldots,\bar{x}_n^I,\underline{x}_1^F,\ldots,\underline{x}_n^F,\bar{x}_1^F,\ldots,\bar{x}_n^F\right)^T$ and $b_i^{\aleph} = \left(\underline{b}_1^T,\ldots,\underline{b}_n^T,\bar{b}_1^T,\ldots,\bar{b}_n^T,\underline{b}_1^I,\ldots,\underline{b}_n^I,\bar{b}_1^I,\ldots,\bar{b}_n^I,\underline{b}_1^F,\ldots,\underline{b}_n^F,\bar{b}_1^F,\ldots,\bar{b}_n^F\right)^T$, following the methodology outlined in Friedman et al. [17], then a $6n\times 6n$ crisp linear system must be solved as:

$$MX = B \tag{7}$$

where,

$$M = \begin{bmatrix} S_{2n\times 2n} & [0]_{2n\times 2n} & [0]_{2n\times 2n} \\ [0]_{2n\times 2n} & S_{2n\times 2n} & [0]_{2n\times 2n} \\ [0]_{2n\times 2n} & [0]_{2n\times 2n} & S_{2n\times 2n} \end{bmatrix}, B = \begin{bmatrix} B^T \\ B^I \\ B^F \end{bmatrix}. \tag{8}$$

In addition, $S = (s_{ij})$, and the following can be obtained:

$$\begin{cases} a_{ij} \geq 0 \rightarrow s_{ij} = a_{ij}, s_{i+n,j+n} = a_{ij} \\ a_{ij} < 0 \rightarrow s_{i,j+n} = -a_{ij}, s_{i+n,j} = -a_{ij} \end{cases} \tag{9}$$

Any s_{ij} which is not determined by Eq. (9) is zero. Therefore:

$$S = \begin{bmatrix} S_1 & -S_2 \\ -S_2 & S_1 \end{bmatrix}, B^T = \begin{bmatrix} b^T \\ \overline{b^T} \end{bmatrix}, B^I = \begin{bmatrix} b^I \\ \overline{b^I} \end{bmatrix}, B^F = \begin{bmatrix} b^F \\ \overline{b^F} \end{bmatrix}.$$

where, $S_1, S_2 \geq 0$, and $S = S_1 - S_2$.

Since M is a block diagonal matrix, to reduce the computational complexity, the following $2n\times 2n$ crisp linear systems only need to be solved:

$$Sx^i = B^i, i = T, I, F \tag{10}$$

Next, following the methodology outlined in Friedman et al. [17], some theorems regarding the properties of S were studied.

Theorem 1. S is nonsingular if $A = S_1 + S_2$ and $S_1 - S_2$ are nonsingular.

Theorem 2. If $S^{(-1)}$ exists, it must have the same structure as S, i.e,

$$S^{-1} = \begin{pmatrix} E & F \\ F & E \end{pmatrix}$$

Definition 8. Let $x_i^{\aleph} = \left(\underline{x}_1^T,\ldots,\underline{x}_n^T,\bar{x}_1^T,\ldots,\bar{x}_n^T,\underline{x}_1^I,\ldots,\underline{x}_n^I,\bar{x}_1^I,\ldots,\bar{x}_n^I,\underline{x}_1^F,\ldots,\underline{x}_n^F,\bar{x}_1^F,\ldots,\bar{x}_n^F\right)^T$ be the unique solution of Eq. (4). If $\forall k \in \{1,2,\ldots,n\} : \underline{x}_k^T \leq \bar{x}_k^T, \underline{x}_k^I \leq \bar{x}_k^I$ and $\underline{x}_k^F \leq \bar{x}_k^F$, then the solution $x_i^{\aleph}$ is called a strong neutrosophic solution. Otherwise, it is a weak neutrosophic solution.

Theorem 3. It is assumed that $S = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_1 \end{bmatrix}$ is a nonsingular matrix. Then Eq. (4) has a strong solution if:

$$(S_1 - S_2)^{-1}\left(\underline{b}^i - \bar{b}^i\right) \leq 0, i = T, I, F \tag{11}$$

Proof:

From Eq. (10), the following can be obtained:

$$\begin{pmatrix} S_1 & S_2 \\ S_2 & S_1 \end{pmatrix} \begin{pmatrix} \underline{x}^i \\ \bar{x}^i \end{pmatrix} = \begin{pmatrix} \underline{b}^i \\ \bar{b}^i \end{pmatrix}, i = T, I, F$$

Hence,

$$S_1 \underline{x}^i - S_2 \bar{x}^i = \underline{b}^i \tag{12}$$

$$-S_2 \underline{x}^i + S_1 \bar{x}^i = \bar{b}^i \tag{13}$$

From Eqs. (12) and (13), the following can be obtained:

$$\begin{cases} (S_1 + S_2)\, \underline{x}^i - (S_1 + S_2)\, \bar{x}^i = \underline{b}^i - \bar{b}^i, \\ (S_1 + S_2)\, (\underline{x} - \bar{x}) = \underline{b}^i - \bar{b}^i. \end{cases}$$

From Theorem 1, $S_1 - S_2$ is nonsingular. Therefore,

$$\left(\underline{x}^i - \bar{x}^i\right) = (S_1 - S_2)^{-1} \left(\underline{b}^i - \bar{b}^i\right) \tag{14}$$

If Eq. (4) has a strong solution, then Eq. (11) holds by Definition 5 $\left(\underline{x}^i - \bar{x}^i \leq 0\right)$. Conversely, if Eq. (11) holds, $\left(\underline{x}^i - \bar{x}^i \leq 0\right)$ can be obtained by Eq. (14).

By Theorems 1 and 3, the result can be obtained below.

Theorem 4. The SNLE has a strong solution if the following conditions hold:

(a) The matrices $A = S_1 + S_2$ and $S_1 - S_2$ are both nonsingular.

(b) $(S_1 - S_2)^{-1} \left(\underline{b}^i - \bar{b}^i\right) \leq 0$.

3.2 Gradient Descent Method for Solving SNLE

In this section, the solution to the SNLE was approximated numerically using the gradient descent method implemented in PyTorch. It is important to note that while embedding methods compute exact solutions, they often become computationally expensive for large-scale or parameterized systems. Iterative methods, such as gradient descent, present scalable and robust alternatives by leveraging numerical optimization techniques to approximate solutions.

The parameterized system $Ax^{\aleph} = B^{\aleph}$ was solved using gradient descent, where A is a fixed coefficient matrix and $B^{\aleph}$ is parameterized by (α, β, γ), introducing controlled variability to evaluate the optimization process under different configurations.

The description of the algorithm is as follows:

(a) Input:

- Matrix $A \in \mathbb{R}^{m \times n}$
- Vector $B^N \in \mathbb{R}^{m \times 1}$
- Initial guess $x^0 \in \mathbb{R}^{n \times 1}$
- Learning rate $\eta > 0$
- Stopping threshold $\varepsilon > 0$

(b) Output:

- Optimized parameter vector $\hat{x}$

The steps of the algorithm are as follows:

Step 1: Initialization by setting $\hat{x} = x^0$.

Step 2: Iterative optimization. The following steps were repeated until the stopping criterion was met:

(a) Residual calculation by computing the residual $\Delta = A \cdot x - B$

(b) Loss function evaluation by computing the loss $L = \|\Delta\|_2$

(c) Gradient calculation by computing the gradient of $L(x)$ with respect to x. PyTorch's automatic differentiation was used to compute the gradient of L with respect to x:

$$\nabla L(x) = 2A^T (A \cdot x - B)$$

(d) Parameter update. x was updated by using the gradient descent rule:

$$x^{k+1} = x^k - \eta \cdot \nabla L(x)$$

where, η is the learning rate controlling the step size.

(e) Stopping criterion. The process terminated when $L(x)$ fell below a predefined threshold ε.

Step 3: Result output by returning $\hat{x} = x$.

The optimization loop was implemented in PyTorch as follows.

The below following flowchart summarizes the steps needed to solve the SVTrNLF (Figure 1).

```
import torch
# Inputs: A, B, initial_x, step_size, epsilon
x = initial_x.clone().detach().requires_grad_(True) # Ensure gradient tracking
while True:
    # Residual calculation
    delta = torch.matmul(A, x) - B
    # Loss function evaluation
    L = torch.norm(delta, p=2)
    # Compute gradients
    L.backward()
    # Parameter update
    with torch.no_grad():
        x -= step_size * x.grad
        x.grad.zero_()
    # Stopping criterion
    if L.item() < epsilon:
        break
```

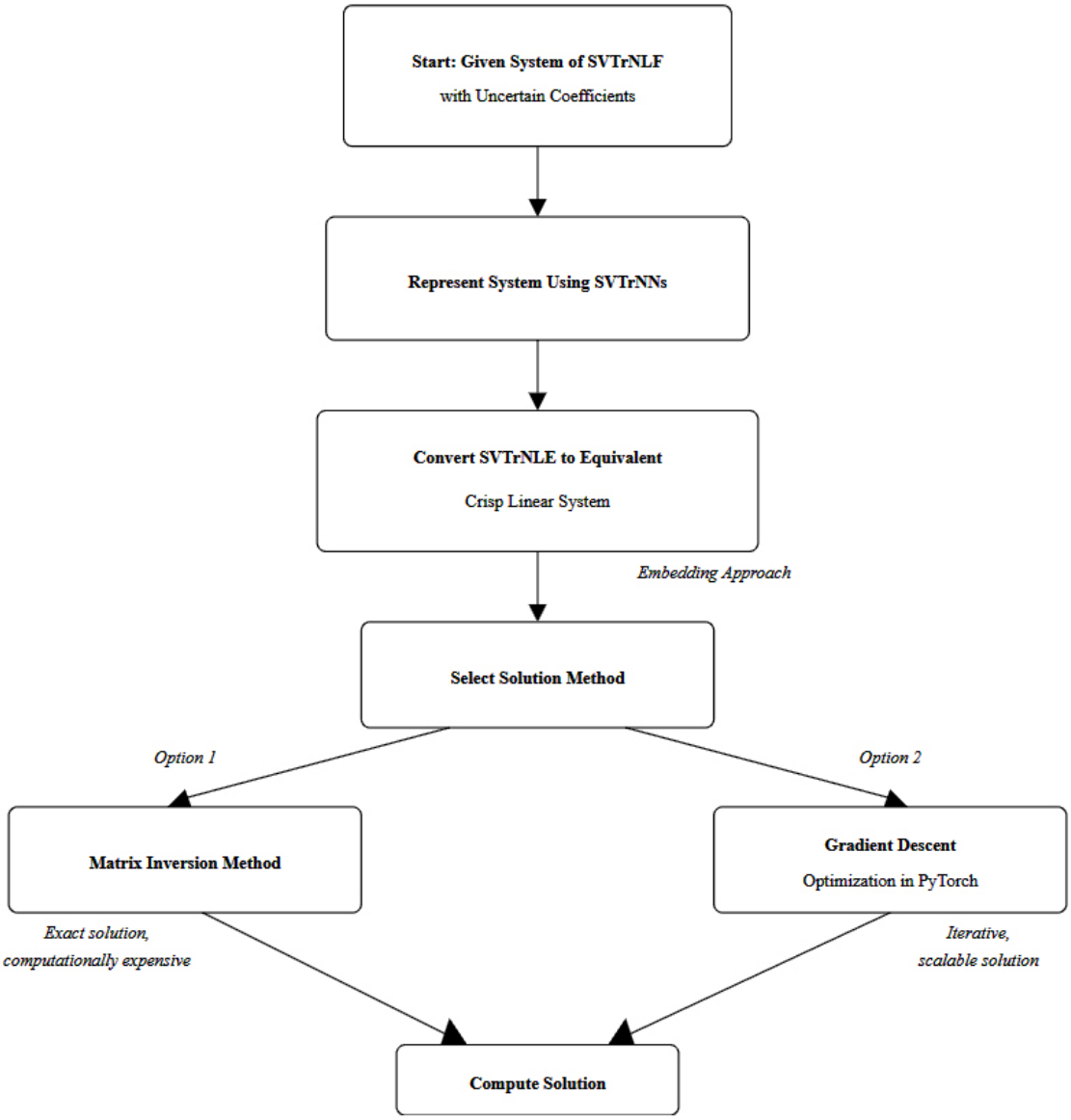

Figure 1. Flowchart for solving SVTrNLF

4 Numerical Experiments

Numerical experiments were conducted to illustrate the results obtained in previous sections.

Example 4.1 By considering the following SNLE:

$$\begin{cases} 4x_1^{\aleph} + x_2^{\aleph} - x_3^{\aleph} =< (1,3,4,5); (0.9,0.3,0.2) > \\ -x_1^{\aleph} + 3x_2^{\aleph} + x_3^{\aleph} =< (0,1,2,3); (0.7,0.4,0.2) > \\ 2x_1^{\aleph} + x_2^{\aleph} + 3x_3^{\aleph} =< (3,5,7,8); (0.8,0.3,0.3) > \end{cases} \tag{15}$$

The extended 4×4 matrix is as follows:

$$S = \begin{bmatrix} 4 & 1 & 0 & 0 & 0 & -1 \\ 0 & 3 & 1 & -1 & 0 & 0 \\ 2 & 1 & 3 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 & 1 & 0 \\ -1 & 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 2 & 1 & 3 \end{bmatrix}$$

Since the matrices $A = S_1 + S_2$ and $S_1 - S_2 = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 3 & 1 \\ 2 & 1 & 3 \end{bmatrix}$ are both nonsingular, then by Theorem 1, it is easy to see that the matrix S is nonsingular. Therefore, $S^{(-1)}$ exists and based on Theorem 2, it must have the same structure as S. If this inverse can be obtained, it can be seen that Theorem 2 is true.

$$S^{-1} = \begin{bmatrix} \frac{35}{143} & \frac{-12}{143} & \frac{1}{143} & \frac{-9}{143} & \frac{-1}{143} & \frac{12}{143} \\ \frac{43}{1144} & \frac{201}{572} & \frac{-105}{1144} & \frac{87}{1144} & \frac{-19}{572} & \frac{27}{1144} \\ \frac{-201}{1144} & \frac{-35}{572} & \frac{411}{1144} & \frac{19}{1144} & \frac{9}{572} & \frac{-73}{1144} \\ \frac{-9}{143} & \frac{-1}{143} & \frac{12}{143} & \frac{35}{143} & \frac{-12}{143} & \frac{1}{143} \\ \frac{87}{1144} & \frac{-19}{572} & \frac{27}{1144} & \frac{43}{1144} & \frac{201}{572} & \frac{-105}{1144} \\ \frac{19}{1144} & \frac{9}{572} & \frac{-73}{1144} & \frac{-201}{1144} & \frac{-35}{572} & \frac{411}{1144} \end{bmatrix}.$$

The (a,b,g)-cut of the right-hand side vector was obtained. By Definition 4, the following was obtained:

$$\begin{aligned} b^{\mathrm{N}}_{1(\alpha,\beta,\gamma)} &= \langle [(1+\alpha(3-1))0.9, (5-\alpha(5-4))0.9], [(3-\beta(3-1))0.3, (5+\beta(5-4))0.3], \\ &\quad [(3-\gamma(3-1))0.2, (5+\gamma(5-4))0.2] \rangle, \\ b^{\mathrm{N}}_{(\alpha,\beta,\gamma)} &= \langle [(0+\alpha(1-0))0.7, (3-\alpha(3-2))0.7], [(1-\beta(1-0))0.4, (3+\beta(3-2))0.4], \\ &\quad [(1-\gamma(1-0))0.2, (3+\gamma(3-2))0.2] \rangle, \\ b^{\mathrm{N}}_{3}(\alpha,\beta,\gamma) &= \langle [(3+\alpha(5-3))0.8, (8-\alpha(8-7))0.8], [(5-\beta(5-3))0.3, (8+\beta(8-7))0.3], \\ &\quad [(5-\gamma(5-3))0.3, (8+\gamma(8-7))0.3] \rangle. \end{aligned}$$

In addition,

$$(S_1+S_2)^{-1} = \begin{bmatrix} 0.18181818 & -0.09090909 & 0.09090909 \\ 0.11363636 & 0.31818182 & -0.06818182 \\ -0.15909091 & -0.04545455 & 0.29545455 \end{bmatrix}.$$

Therefore:

$$\begin{aligned} (S_1+S_2)^{-1}\left(\underline{b}^T - \bar{b}^T\right) &= \begin{bmatrix} 0.18181818 & -0.09090909 & 0.09090909 \\ 0.11363636 & 0.31818182 & -0.06818182 \\ -0.15909091 & -0.04545455 & 0.29545455 \end{bmatrix} \begin{bmatrix} 2.7\alpha - 3.5 \\ 1.4\alpha - 2.1 \\ 2.4\alpha - 3.4 \end{bmatrix} \\ &= \begin{bmatrix} 0.581818176 \cdot \alpha - 0.754545447 \\ 0.588636352 \cdot \alpha - 0.834090894 \\ 0.215909093 \cdot \alpha - 0.35227273 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\ (S_1+S_2)^{-1}\left(\underline{b}^I - \bar{b}^I\right) &= \begin{bmatrix} 0.18181818 & -0.09090909 & 0.09090909 \\ 0.11363636 & 0.31818182 & -0.06818182 \\ -0.15909091 & -0.04545455 & 0.29545455 \end{bmatrix} \begin{bmatrix} -2 - 0.9\beta \\ -2 - 0.8\beta \\ -2.4 - 0.9\beta \end{bmatrix} \\ &= \begin{bmatrix} -0.172727271 \cdot \beta - 0.399999996 \\ -0.295454542 \cdot \beta - 0.699999992 \\ -0.086363636 \cdot \beta - 0.3 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \end{aligned}$$

$$(S_1+S_2)^{-1}\left(\underline{b}^F-\bar{b}^F\right)=\begin{bmatrix}0.18181818 & -0.09090909 & 0.09090909\\ 0.11363636 & 0.31818182 & -0.06818182\\ -0.15909091 & -0.04545455 & 0.29545455\end{bmatrix}\begin{bmatrix}-0.4-0.6\gamma\\ -0.4-0.4\gamma\\ -0.9-0.9\gamma\end{bmatrix}$$
$$=\begin{bmatrix}-0.154545453\cdot\gamma-0.118181817\\ -0.134090906\cdot\gamma-0.111363634\\ -0.152272729\cdot\gamma-0.184090911\end{bmatrix}\leq\begin{bmatrix}0\\0\end{bmatrix},$$

Therefore, by Theorems 3 and 4, the SVTrNLEs in Eq. (15) should have a strong solution. To obtain this solution, the following was obtained from Eq. (9):

$$x^T=\begin{bmatrix}\underline{x}_1^T\\ \underline{x}_2^T\\ \bar{x}_1^T\\ \bar{x}_2^T\end{bmatrix}=S^{-1}B^T=\begin{bmatrix}0.387412587412587\cdot\alpha+0.476223776223776\\ 0.10270979020979\cdot\alpha+0.237062937062937\\ 0.240821678321678\cdot\alpha+0.403496503496504\\ 1.11468531468531-0.151048951048951\cdot\alpha\\ 0.444755244755245-0.0549825174825175\cdot\alpha\\ 1.24195804195804-0.14763986013986\cdot\alpha\end{bmatrix},$$

$$x^I=\begin{bmatrix}\underline{x}_1^I\\ \underline{x}_2^I\\ \bar{x}_1^I\\ \bar{x}_2^I\end{bmatrix}=S^{-1}B^I=\begin{bmatrix}0.295804195804196-0.113986013986014\cdot\beta\\ 0.16756993006993-0.0914335664335664\cdot\beta\\ 0.24694055944056-0.0935314685314685\cdot\beta\\ 0.0321678321678322\cdot\beta+0.34965034965035\\ 0.0777972027972028\cdot\beta+0.348339160839161\\ 0.0526223776223776\cdot\beta+0.450786713286713\end{bmatrix},$$

$$x^F=\begin{bmatrix}\underline{x}_1^F\\ \underline{x}_2^F\\ \bar{x}_1^F\\ \bar{x}_2^F\end{bmatrix}=S^{-1}B^F=\begin{bmatrix}0.274825174825175-0.0741258741258741\cdot\gamma\\ 0.0679195804195804-0.0145979020979021\cdot\gamma\\ 0.294143356643357-0.145716783216783\cdot\gamma\\ 0.0104895104895105\cdot\gamma+0.297902097902098\\ 0.0123251748251748\cdot\gamma+0.102534965034965\\ 0.0888986013986014\cdot\gamma+0.56722027972028\end{bmatrix}.$$

Therefore,

$$\begin{aligned}x^{\aleph}_{1(\alpha,\beta,\gamma)}&=\langle[0.387412587412587\cdot\alpha+0.476223776223776,1.11468531468531-0.151048951048951\cdot\alpha],\\ &\quad[0.295804195804196-0.113986013986014\cdot\beta,0.0321678321678322\cdot\beta+0.34965034965035],\\ &\quad[0.274825174825175-0.0741258741258741\cdot\gamma,0.0104895104895105\cdot\gamma+0.297902097902098]\rangle,\\ x^{\aleph}_{2(\alpha,\beta,\gamma)}&=\langle[0.10270979020979\cdot\alpha+0.237062937062937,0.444755244755245-0.0549825174825175\cdot\alpha],\\ &\quad[0.16756993006993-0.0914335664335664\cdot\beta,0.0777972027972028\cdot\beta+0.348339160839161],\\ &\quad[0.0679195804195804-0.0145979020979021\cdot\gamma,0.0123251748251748\cdot\gamma+0.102534965034965]\rangle,\\ x^{\mathrm{N}}_{3^{\aleph}_{(\alpha,\beta,\gamma)}}&=\langle[0.240821678321678\cdot\alpha+0.403496503496504,1.24195804195804-0.14763986013986\cdot\alpha],\\ &\quad[0.24694055944056-0.0935314685314685\cdot\beta,0.0526223776223776\cdot\beta+0.450786713286713],\\ &\quad[0.294143356643357-0.145716783216783\cdot\gamma,0.0888986013986014\cdot\gamma+0.56722027972028]\rangle,\end{aligned}$$

For different values of $0\leq\alpha,\beta,\gamma\leq 1$, the graphical interpretation of the above results is shown in Figures 2, 3 and 4.

5 Results

This section analyzes the convergence behavior of gradient descent for parameterized linear systems and evaluates the accuracy of the computed solutions by comparing them to the exact solutions. Additionally, the robustness and stability of gradient descent were examined under varying parameter configurations. As a foundational technique in the training of neural networks, gradient descent is employed to minimize error functions in high-dimensional parameter spaces [40]. This study investigates the behavior of gradient descent within the context of simpler linear systems, offering insights that contribute to understanding its application in more complex scenarios.

5.1 Convergence Behavior

Figures 5, 6, and 7 show the convergence behavior of gradient descent for different parameter configurations.

As shown in Figure 5, loss starts at approximately 8.5 and converges to near zero within 139 iterations. The consistent decrease in loss demonstrates stability and effective learning.

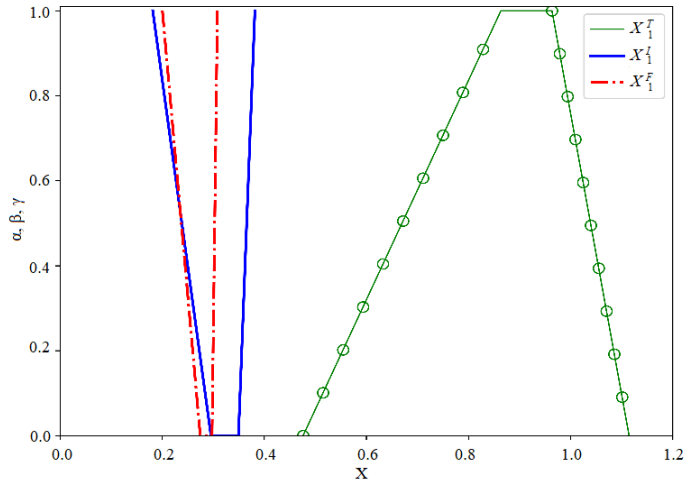

Figure 2. The value of $x_1^{\mathcal{H}}$

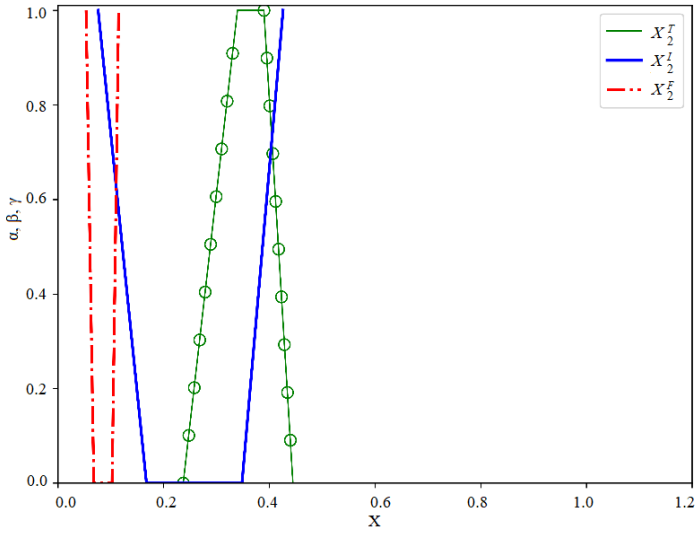

Figure 3. The value of $x_2^{\mathcal{H}}$

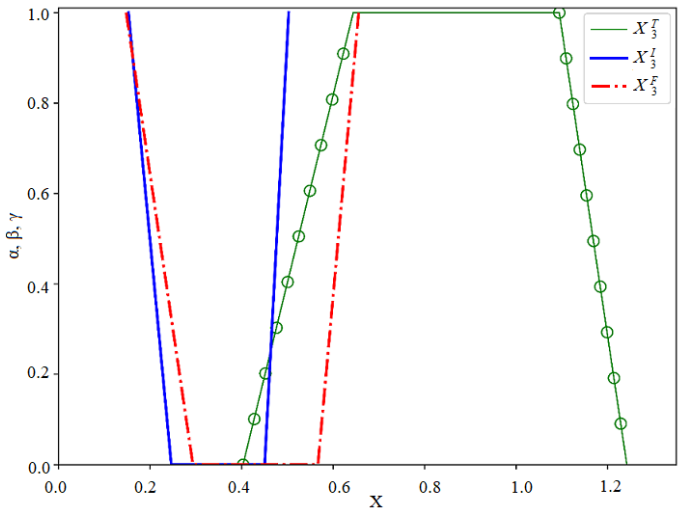

Figure 4. The value of $x_3^{\mathcal{H}}$

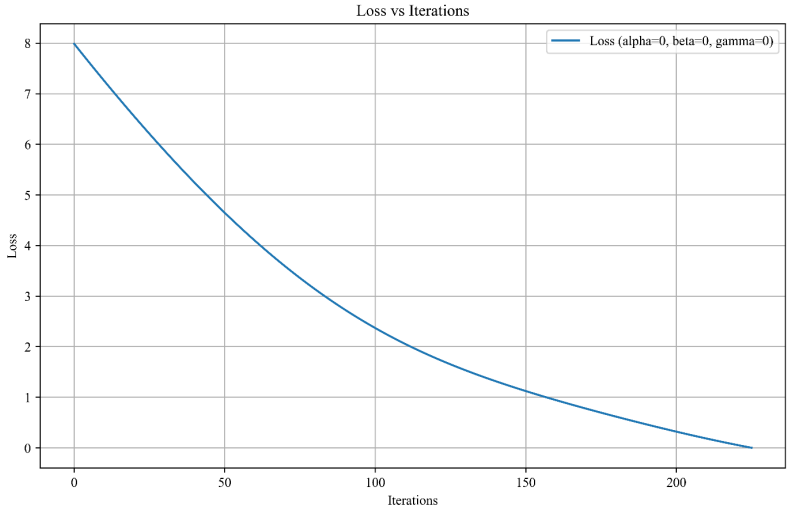

Figure 5. Loss vs. iterations (α=0, β=0, γ=0)

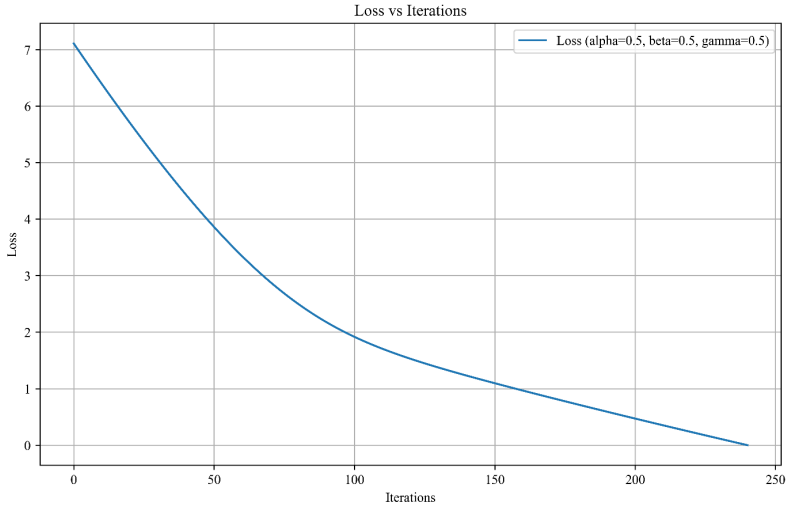

Figure 6. Loss vs. iterations (α=0.5, β=0.5, γ=0.5)

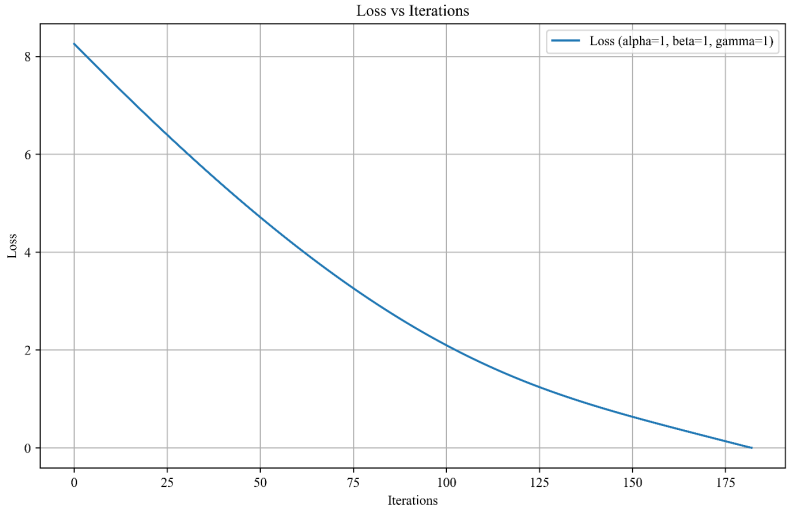

Figure 7. Loss vs. iterations (α=1, β=1, γ=1)

As shown in Figure 6, loss converges slightly more slowly, taking 156 iterations to reach a threshold below $10^{(-2)}$. The increase in parameter variability introduces more complexity to the optimization process.

As shown in Figure 7, loss decreases rapidly, converging within 171 iterations. This suggests a favorable alignment between the parameterized B and the structure of A. Table 1 shows the comparison results for Example 4.1.

Table 1. Comparison results for Example 4.1

	Alph=0	**0.5**	**1**
Iteration to converge	139	156	171
Loss at convergence	0.006949843838810921	0.002292076358571648	0.0011400537332519889
Final loss	0.004616653081029653 5	0.010320821776986122	0.015670282766222954
Computed solutions	$\begin{pmatrix} 0.4765 & 0.2953 & 0.2750 \\ 0.2374 & 0.1670 & 0.0682 \\ 0.4028 & 0.2482 & 0.2935 \\ 1.1144 & 0.3501 & 0.2977 \\ 0.4444 & 0.3489 & 0.1023 \\ 1.2426 & 0.4496 & 0.5678 \end{pmatrix}$	$\begin{pmatrix} 0.6703 & 0.2396 & 0.2382 \\ 0.2881 & 0.1233 & 0.0599 \\ 0.5234 & 0.1977 & 0.2215 \\ 1.0389 & 0.3650 & 0.3027 \\ 0.4167 & 0.3855 & 0.1101 \\ 1.1695 & 0.4799 & 0.6108 \end{pmatrix}$	$\begin{pmatrix} 0.8642 & 0.1819 & 0.1995 \\ 0.3409 & 0.0769 & 0.0551 \\ 0.6402 & 0.1524 & 0.1481 \\ 0.9627 & 0.3818 & 0.3093 \\ 0.3913 & 0.4246 & 0.1156 \\ 1.0958 & 0.5051 & 0.6541 \end{pmatrix}$
Exact solutions	$\begin{pmatrix} 0.4762 & 0.2958 & 0.2748 \\ 0.2371 & 0.1676 & 0.0679 \\ 0.4035 & 0.2469 & 0.2941 \\ 1.1147 & 0.3496 & 0.2979 \\ 0.4448 & 0.3483 & 0.1025 \\ 1.2420 & 0.4508 & 0.5672 \end{pmatrix}$	$\begin{pmatrix} 0.6699 & 0.2388 & 0.2378 \\ 0.2884 & 0.1219 & 0.0606 \\ 0.5239 & 0.2002 & 0.2213 \\ 1.0392 & 0.3657 & 0.3031 \\ 0.4173 & 0.3872 & 0.1087 \\ 1.1681 & 0.4771 & 0.6117 \end{pmatrix}$	$\begin{pmatrix} 0.8636 & 0.1818 & 0.2007 \\ 0.3398 & 0.0761 & 0.0533 \\ 0.6443 & 0.1534 & 0.1484 \\ 0.9636 & 0.3818 & 0.3084 \\ 0.3898 & 0.4261 & 0.1149 \\ 1.0943 & 0.5034 & 0.6561 \end{pmatrix}$
Distances from exact solutions	0.0024775145575404167	0.005190060473978519	0.006468791514635086

5.2 Accuracy Analysis

The computed solutions were compared to the exact solutions. The Euclidean distances between the computed solution and the exact solution were:

- Case $1(\alpha = 0, \beta = 0, \gamma = 0) : 0.0012$
- Case $2(\alpha = 0.5, \beta = 0.5, \gamma = 0.5) : 0.0063$
- Case $3(\alpha = 1, \beta = 1, \gamma = 1) : 0.0014$

The small distances demonstrate the precision of gradient descent, with minimal deviations from the exact solutions.

6 Discussion

6.1 Learning Dynamics

The gradient descent process showed smooth and predictable convergence behavior. The rapid reduction in loss across all cases highlights the efficiency of the method. Differences in convergence speed reflect the influence of parameterized variability in B, with higher parameter values leading to faster alignment with the system's structure.

6.2 Accuracy

The computed solutions x closely matched the exact solutions X_{exact}. The small distances, consistently below 0.01, validate the method's accuracy, even for complex configurations of B.

6.3 Numerical Stability

The loss consistently decreased without oscillations, demonstrating numerical stability. The choice of learning rate and stopping criterion ensured efficient and stable optimization.

6.4 Gradient Descent in Practice

The results illustrate that gradient descent effectively balances computational efficiency and precision. Its iterative nature and adaptability make it suitable for large-scale or parameterized systems where exact methods may be impractical.

7 Conclusions

In conclusion, this study provides a comprehensive examination of SVTrNLEs as an effective framework for addressing uncertainty and imprecision in linear systems. Through the embedding approach, SVTrNLEs were transformed into equivalent crisp linear systems, enabling the use of classical solution techniques. The study rigorously investigates the existence and uniqueness of the solution to an $n \times n$ linear system and introduces a computational method for solving SVTrNLEs. Numerical experiments validate the reliability and efficiency of the proposed method, underscoring its capability to handle systems with indeterminate and vague parameters effectively. Additionally, this study demonstrates the robustness and accuracy of gradient descent in solving parameterized linear systems. By iteratively minimizing the residual error, the method achieved high precision across diverse configurations of BB. These results emphasize the potential of gradient descent for solving complex systems in science and engineering, bridging the gap between theoretical optimization and practical applications. Future research could focus on developing more efficient algorithms for solving larger and more complex neutrosophic systems, further advancing the applicability of these approaches in real-world scenarios.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] D. C. Lay, S. R. Lay, and J. J. McDonald, *Linear Algebra and Its Application*, 6th ed. Pearson, 2020. https://www.pearson.com/en-us/subject-catalog/p/linear-algebra-and-its-applications/P200000006235/9780136880929

[2] D. Kincaid and W. Cheney, "Numerical analysis–Mathematics of scientific computing," *Math. Comput.*, vol. 59, no. 199, pp. 297–298, 1992. https://doi.org/10.2307/2152998

[3] S. A. Edalatpanah, "An experimental comparison of two preconditioned iterative methods to solve the elliptic partial differential equations," *Comput. Algorithms Numer. Dimens.*, vol. 1, no. 1, pp. 1–24, 2022. https://doi.org/10.22105/cand.2022.155122

[4] M. Shams and N. Kausar, "On efficient iterative schemes for finding all solutions of non-linear engineering problems," *Comput. Algorithms Numer. Dimens.*, vol. 3, no. 3, pp. 201–207, 2024. https://doi.org/10.22105/cand.2024.474645.1103

[5] M. Lamtar-Gholipoor, S. Fakheri, and M. Alimoradi, "Artificial neural network TSR for optimization of actinomycin production," *Big Data Comput. Vis.*, vol. 4, no. 1, pp. 57–66, 2024. https://doi.org/10.22105/bdcv.2024.474793.1184

[6] R. E. Miller and P. D. Blair, *Input-Output Analysis: Foundations and Extensions*, 2nd ed. Cambridge University Press, 2009. https://doi.org/10.1017/CBO9780511626982

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, 2003.

[9] A. Greenbaum, *Iterative Methods for Solving Linear Systems*. Frontiers in Applied Mathematics, 1997. https://doi.org/10.1137/1.9781611970937

[10] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, 25th ed. Other Titles in Applied Mathematics, 2022. https://doi.org/10.1137/1.9781611977165

[11] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965. https://doi.org/10.1016/S0019-9958(65)90241-X

[12] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-I," *Inf. Sci.*, vol. 8, no. 3, pp. 199–249, 1975. https://doi.org/10.1016/0020-0255(75)90036-5

[13] A. Kaufmann and M. M. Gupta, "Introduction to fuzzy arithmetic, theory and applications," *Math. Comput.*, vol. 47, no. 176, pp. 762–763, 1986. https://doi.org/10.2307/2008199

[14] A. Kaufmann and M. Gupta, *Introduction to Fuzzy Arithmetic*. Van Nostrand Reinhold Company, 1991.

[15] H. J. Zimmermann, *Fuzzy Set Theory—and Its Applications*. Springer, 2011.

[16] D. J. Dubois, *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, 1980.

[17] M. Friedman, M. Ming, and A. Kandel, "Fuzzy linear systems," *Fuzzy Sets Syst.*, vol. 96, no. 2, pp. 201–209, 1998. https://doi.org/10.1016/S0165-0114(96)00270-9

[18] S. Abbasbandy and A. Jafarian, "Steepest descent method for a system of fuzzy linear equations," *Appl. Math. Comput.*, vol. 175, no. 1, pp. 823–833, 2006. https://doi.org/10.1016/j.amc.2005.07.036

[19] F. Abbasi and T. Allahviranloo, "Solving fully fuzzy linear system: A new solution concept," *Inf. Sci.*, vol. 589, pp. 608–635, 2022. https://doi.org/10.1016/j.ins.2022.01.004

[20] T. Allahviranloo, "Numerical methods for fuzzy system of linear equations," *Appl. Math. Comput.*, vol. 155, no. 2, pp. 493–502, 2004. https://doi.org/10.1016/S0096-3003(03)00793-8

[21] T. Allahviranloo, M. Ghanbari, A. A. Hosseinzadeh, E. Haghi, and R. Nuraei, "A note on "fuzzy linear systems"," *Fuzzy Sets Syst.*, vol. 177, no. 1, pp. 87–92, 2011. https://doi.org/10.1016/j.fss.2011.02.010

[22] X. Guo and Q. Zhuo, "Perturbation analysis of fully fuzzy linear systems," *J. Intell. Fuzzy Syst.*, vol. 44, no. 4, pp. 5589–5599, 2023. https://doi.org/10.3233/JIFS-222392

[23] G. Malkawi, N. Ahmad, and H. Ibrahim, "A note on "The nearest symmetric fuzzy solution for a symmetric fuzzy linear system"," *An. St. Univ. Ovidius Constanta*, vol. 23, no. 2, pp. 173–177, 2014. https://doi.org/10.1515/auom-2015-0034

[24] N. Mikaeilvand, "On solvability of fuzzy system of linear matrix equations," *J. Appl. Sci. Res.*, vol. 7, no. 2, pp. 141–153, 2011.

[25] S. H. Nasseri and H. Attari, "Chebyshev acceleration technique for solving fuzzy linear system," *Iran. J. Optim.*, vol. 3, no. 2, pp. 247–256, 2011.

[26] J. F. Yin and K. Wang, "Splitting iterative methods for fuzzy system of linear equations," *Comput. Math. Model.*, vol. 20, pp. 326–335, 2009. https://doi.org/10.1007/s10598-009-9039-9

[27] G. Malkawil, E. Adnan, H. Albarghouthi, and A. Elsayed, "The neither unique nor infinite solution of fully fuzzy linear system for triangular fuzzy numbers (m, α, β)," *J. Adv. Res. Dyn. Control Syst.*, vol. 11, no. 12, pp. 267–274, 2019.

[28] A. A. Elsayed, N. Ahmad, and G. Malkawi, "Arbitrary generalized trapezoidal fully fuzzy sylvester matrix equation," *Int. J. Fuzzy Syst. Appl.*, vol. 11, no. 1, pp. 1–22, 2022. https://doi.org/10.4018/IJFSA.303564

[29] A. A. Elsayed, N. Ahmad, and G. Malkawi, "Numerical solutions for coupled trapezoidal fully fuzzy sylvester matrix equations," *Adv. Fuzzy Syst.*, vol. 2022, p. 8926038, 2022. https://doi.org/10.1155/2022/8926038

[30] A. A. A. Elsayed, N. Ahmad, and G. Malkawi, "Solving positive trapezoidal fully fuzzy sylvester matrix equation," *Fuzzy Inf. Eng.*, vol. 14, no. 3, pp. 314–334, 2023. https://doi.org/10.1080/16168658.2022.2152906

[31] K. T. Atanassov, "Intuitionistic fuzzy sets," *Fuzzy Sets Syst.*, vol. 20, no. 1, pp. 87–96, 1986. https://doi.org/10.1016/S0165-0114(86)80034-3

[32] F. Smarandache, *A Unifying Field in Logics: Neutrosophic Logic*. American Research Press, 1999.

[33] Y. A. Alhasan, "Types of system of the neutrosophic linear equations and cramer's rule," *Neutrosophic Sets Syst.*, vol. 45, no. 1, pp. 402–413, 2021.

[34] S. A. Edalatpanah, "Systems of neutrosophic linear equations," *Neutrosophic Sets Syst.*, vol. 33, no. 1, pp. 92–104, 2020.

[35] S. A. Edalatpanah, "General non-square systems of linear equations in single-valued triangular neutrosophic number environment," in *Neutrosophic Theories in Communication, Management and Information Technology*. New York, Nova Science Publishers, 2020, pp. 157–172.

[36] M. Jdid and F. Smarandache, "A study of systems of neutrosophic linear equations," *Int. J. Neutrosophic Sci.*, vol. 23, no. 2, pp. 16–25, 2024. https://doi.org/10.54216/IJNS.230202

[37] J. Ye, "Neutrosophic linear equations and application in traffic flow problems," *Algorithms*, vol. 10, no. 4, p. 133, 2017. https://doi.org/10.3390/a10040133

[38] J. Ye, "Multicriteria decision-making method using the correlation coefficient under single-valued neutrosophic environment," *Int. J. Gen. Syst.*, vol. 42, no. 4, pp. 386–394, 2013. https://doi.org/10.1080/03081079.2012.761609

[39] M. R. Seikh and S. Dutta, "Solution of matrix games with payoffs of single-valued trapezoidal neutrosophic numbers," *Soft Comput.*, vol. 26, pp. 921–936, 2022. https://doi.org/10.1007/s00500-021-06559-7

[40] J. Heaton, "Ian goodfellow, yoshua bengio, and aaron courville: Deep learning," *Genet. Program. Evolvable Mach.*, vol. 19, no. 1, pp. 305–307, 2018. https://doi.org/10.1007/s10710-017-9314-z