International Journal of Applied Mathematics

Volume 38 No. 9s , 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Estimated Optimal Parameter for Comparing Methods in a Non-Symmetric Fuzzy Linear System

Al-Shimari, Nofl Sh[1],Ahmed Abdulhussein Jabbar [2]
,Mohammed Shakir Mahdi Zabiba [3]
[1] Ministry of Education
General Directorate of Education in Babil / Iraqs e-mail: noflshaker8@gmail.com
[2] Ministry of Education
General Directorate of Education Najaf / Iraq
ahmed.jabbar.pure246@student.uobabylon.edu.iq
[3] Iraq - University of Kufa
Faculty of Education for Women -Department of Mathematics
mohammedsh.mahdi@uokufa.edu.iq

Abstract

This paper emphasizes the crucial importance of up- date strategies and relaxation parameters in improving com- putational efficiency, offering essential guidance for select- ing suitable solvers in fuzzy linear algebra applications.The Conjugate Gradient (*CG*) method is employed to assess the spectral radius of the Jacobi-preconditioned operator, which aids in determining the optimal relaxation factor .The sys- tem is examined through parametric formulations at differ- ent -cuts to effectively handle uncertainty limits. Conver- gence is evaluated based on a strict residual tolerance crite- rion, with findings revealing significant variations in compu- tational efficiency among the methods.The Jacobi method exhibited a slower convergence rate, while the Gauss-Seidel method improved performance through sequential updates. The Successive Over-Relaxation (*SOR*) method, fine-tuned with a carefully adjusted relaxation factor, achieved the fastest convergence, underscoring the importance of param- eter optimization. All methods yielded consistent fuzzy so- lutions, thereby confirming the numerical reliability and ro- bustness of the iterative approach. This study conducts a comparative analysis of iterative methods-Jacobi, Gauss- Seidel, and SOR-for solving a non-symmetric fuzzy linear system defined by triangular fuzzy coefficients and constants. **Key Words** non-symmetric fuzzy,the Gauss-Seidel method,
The Conjugate Gradient, (*SOR*) method, Fuzzy Linear Sys- tem.

1 Introduction

Conventional direct methods, such as matrix inversion, are fre- quently impractical for fuzzy systems due to their propensity to amplify errors and their significant computational expense, par- ticularly for larger systems [3]. Iterative techniques like Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR) provide ro- bust alternatives, effectively managing fuzzy arithmetic through sequential approximations [4]. Fuzzy linear systems, characterized by coefficients and variables represented as fuzzy numbers, are vital for modeling real-world uncertainties in areas such as engineering design, decision-making, and economic forecasting [1]. The Jacobi method, although straightforward, experiences slow convergence as it depends solely on values from the preceding iteration. In contrast,

Received: August 12 2025

Volume 38 No. 9s , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

the Gauss-Seidel method improves convergence speed by promptly integrating updated values, often significantly decreasing iteration counts in classical linear systems [5]. While symmetric fuzzy sys- tems have been extensively studied, non-symmetric systems-where the spreads of fuzzy coefficients vary independently-pose distinct computational challenges due to their intricate structure [2]. The SOR method, which utilizes a relaxation factor to enhance solu- tions, has demonstrated considerable efficiency gains in determin- istic systems [6]. Addressing these systems requires methods that adeptly manage the dual uncertainty bounds (lower and upper) of triangular fuzzy numbers while ensuring numerical stability and ef- ficiency. Nevertheless, its effectiveness in non-symmetric fuzzy sys- tems, especially in achieving a balance between convergence speed and numerical stability, remains insufficiently explored. Recent ad- vancements in parametric approaches, which decompose fuzzy sys- tems into crisp -cut subproblems, have enabled precise management of uncertainty intervals [7]. This framework supports the dual- solution strategy for lower and upper bounds, which is essential for preserving the accuracy of fuzzy solutions [8]. Despite these developments, comparative studies of iterative methods for non- symmetric fuzzy systems are scarce, leaving gaps in understanding.

2 Evaluating Different Approaches for Linear Fuzzy Systems

Definition 1 (1). Consider the $n \times n$ fully Linear Equations with Fuzzy Parameters:

$$\tilde{a}_{11}\tilde{x}_1 + \tilde{a}_{12}\tilde{x}_2 + \cdots + \tilde{a}_{1n}\tilde{x}_n = \tilde{b}_1$$
$$\tilde{a}_{21}\tilde{x}_1 + \tilde{a}_{22}\tilde{x}_2 + \cdots + \tilde{a}_{2n}\tilde{x}_n = \tilde{b}_2$$
$$\vdots \qquad \vdots$$
$$\tilde{a}_{n1}\tilde{x}_1 + \tilde{a}_{n2}\tilde{x}_2 + \cdots + \tilde{a}_{nn}\tilde{x}_n = \tilde{b}_n$$

The matrix form of the above system is:

$$\tilde{A}\tilde{X} = \tilde{B}$$

This is known as a **Fully Fuzzy Linear System (FFLS)**. This system features a coefficient matrix $\tilde{A}$ of size $n \times n$ with fuzzy number elements, $\tilde{X}$ is the fuzzy vector of unknown variables, and $\tilde{B}$ is the fuzzy vector of right-hand side constants.

2.1 Jacobi and Gauss-Seidel Iterative Methods

The Jacobi and Gauss-Seidel methods are iterative techniques used to solve a linear system of the form $EG = Y$. Starting with an initial approximation $G^{(0)}$ to the solution G, these methods generate a convergent sequence $\{G^{(k)}\}_{k=0}^{\infty}$ that approaches G.
Iterative methods typically transform the system $EG = Y$ into an equivalent form:

$$G = AG + F$$

where A is a matrix (often derived from splitting the coefficient matrix) and F is a vector. Given an initial guess $G^{(0)}$, a sequence of approximations is generated

Received: August 12 2025

Volume 38 No. 9s , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

iteratively using:

$$G^{(k+1)} = AG^{(k)} + F$$

Definition 2 (Diagonally Dominant Matrix)**.** [3] A square matrix E is called **diagonally dominant** if, for each row i, the absolute value of the diagonal element satisfies:

$$|e_{ii}| \geq \sum_{j=1, j\neq i}^{n} |e_{ij}| \quad \text{for } i = 1, 2, \ldots, n.$$

The matrix is **strictly diagonally dominant** if the inequality is strict:

$$|e_{ii}| > \sum_{j=1, j\neq i}^{n} |e_{ij}| \quad \text{for } i = 1, 2, \ldots, n.$$

Theorem 3 (4)**.** *If the matrix E in the system $EG = Y$ is strictly diagonally dominant, both the Jacobi and Gauss-Seidel iterative methods will converge to the solution $E^{-1}Y$ for any initial guess $G^{(0)}$.*

Theorem 4 (4)**.** *If the matrix E in the system $EG = Y$ is strictly diagonally dominant, then the matrix Y (likely referring to a related system or transformed matrix, as per context) is also strictly diagonally dominant.*

2.2 Successive Over-Relaxation Iterative Method

In this section, we shift our focus to an adaptation of the Gauss- Seidel iteration method, known as the **Successive Over-Relaxation (SOR) iterative method**. By applying the SOR iteration to the system described.

Let $\tilde{G}^{(k+1)}$ be the Gauss-Seidel solution, then we can rewrite this as:

$$G^{(k+1)} = G^{(k)} + \omega(\tilde{G}^{(k+1)} - G^{(k)})$$

This reformulated system of equations represents the SOR iterative method, where the matrix coefficients and the right-hand side have been modified compared to the original Gauss-Seidel approach [3]. For some parameter ω, when ω equals 1, it is evident that $G^{(k+1)}$ corresponds to the Gauss-Seidel solution. Consequently, the SOR iterative method can be expressed as follows:

$$a_{ij}g^{(k+1)} + a_{ii}g^{(k+1)} + \sum_{j=1}^{i-1} \quad j \qquad i \qquad \sum_{j=i+1}^{n} a_{ij}g_j^{(k)} = b_i$$

As a result, the outcome presented in matrix form of the SOR iterative method is:

$$G^{(k+1)} = M_\omega^{-1}N_\omega G^{(k)} + M_\omega^{-1}B$$

where

$$M_\omega = D + \omega L \quad \text{and} \quad N_\omega = (1 - \omega)D - \omega U$$

Received: August 12 2025

Volume 38 No. 9s , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

For values of ω ranging from 0 to 1, this technique is referred to as the **successive under-relaxation method**. It is applicable for attaining convergence in systems that do not converge using the Gauss-Seidel method. When ω exceeds 1, the technique is known as the **Successive Over-Relaxation (SOR) method**. This ap- proach can be employed to enhance the convergence speed of lin- ear systems that are already converging through the Gauss-Seidel method [10].

Theorem 5 (5)**.** *If A is a positive definite matrix and* $0 < \omega < 2$*, the Successive Over-Relaxation (SOR) method will converge for any selection of the initial approximate vector* $x^{(0)}$*.*

This conversion translates the provided text into $LaTeX$ code, focusing on the correct formatting of mathematical equations and structures.
"'latex article amsmath, amsfonts, amssymb array Corollary

3 Numerical Approaches for Solving Fully Fuzzy Linear Equation Systems (FFLS)

In the preceding chapters, we have discussed various direct ap- proaches for addressing fully fuzzy linear systems of equations. This section introduces two iterative techniques, specifically the **Gauss- Jacobi** and **Gauss-Seidel** methods, aimed at determining the solutions for fully fuzzy linear systems of equations [12].

3.1 Gauss-Jacobi Method

The system given by the following linear equations has a unique solution:

$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$ $a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$

.

$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$

The coefficient matrix A is characterized by the absence of zeros along its main diagonal, meaning a_{ii} are all non-zero.

Main idea of Jacobi

First, isolate x_1 in the initial equation, then determine x_2 from the subsequent equation, and continue this process to derive the reformulated equations:

$$x_1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n)$$

$$x_2 = \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n)$$

Received: August 12 2025

Volume 38 No. 9s , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

.

$$x_n = \frac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1})$$

Then make an initial guess of the solution $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})^T$.

Replace these values in the right-hand side of the modified equations to derive the initial approximation $x^{(1)}$. This concludes one iteration. The second approximation $x^{(2)}$ is obtained by substitut- ing the values from the first approximation $x^{(1)}$ into the right side of the reformulated equations. By repeating this process, a sequence of approximations is created [7].

The Jacobi method iteratively generates a sequence of approximations. For each iteration $k \geq 1$, the components $x_i^{(k)}$ of the current approximation $x^{(k)}$ are calculated by using the values from the previous approximation $x^{(k-1)}$ and plugging them into the re- formulated equations:

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i} a_{ij} x_j^{(k-1)}\right), \quad \text{excluding } x_i^{(k-1)}.$$

3.1.1 Applying the Jacobi Iterative Algorithm Using Ma- trix Representation

Consider the problem of finding a solution to an n-by-n system of linear equations, represented in matrix form as $Ax = b$, where:

$A = [a_{ij}]$, $x = (x_1, \ldots, x_n)^T$, $b = (b_1, \ldots, b_n)^T$

We split A into:

$A = D + L + U$

Where: D is the diagonal component, L is the strictly lower trian- gular component, and U is the strictly upper triangular component.

The equation $Ax = b$ is transformed into:

$(D + L + U)x = b \Rightarrow Dx = -(L + U)x + b$

Assuming D^{-1} exists and $a_{ii} \neq 0$:

$x = -D^{-1}(L + U)x + D^{-1}b$

The Jacobi iterative method can be expressed in matrix form as:

$x^{(k)} = -D^{-1}(L + U)x^{(k-1)} + D^{-1}b$, where $k \geq 1$.

We can define:

$T_J = -D^{-1}(L + U)$ and $c_J = D^{-1}b$

This allows us to rewrite the Jacobi iteration as:

$x^{(k)} = T_J x^{(k-1)} + c_J$ for $k = 1, 2, \ldots, n$

Received: August 12 2025

International Journal of Applied Mathematics

Volume 38 No. 9s , 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

3.2 The Gauss-Seidel Method

The core concept of the Gauss-Seidel method is to utilize newly computed values immediately, unlike the Jacobi method, which re- lies solely on values from the previous iteration. In the Gauss-Seidel method, the updated values of x_i are applied within the current iter- ation as soon as they are available. For example, the updated value for the first variable, $x^{(k)}$, is first calculated from the first equa- tion. This newly computed $x^{(k)}$ value is then immediately used in the second equation to determine the updated value for the second variable, $x^{(k)}$. This process continues sequentially through all the equations in the system [9].

3.2.1 The Gauss-Seidel Iterative Algorithm

For each $k \geq 1$, generate the components $x^{(k)}$ of $x^{(k)}$ from $x^{(k-1)}$ by:

$$x^{(k)} = \frac{1}{a_{ii}} b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \quad \text{for } i = 1, 2, \ldots n$$

Namely:

$$x_1^{(k)} = \frac{1}{a_{11}} (b_1 - a_{12} x_2^{(k-1)} - a_{13} x_3^{(k-1)} - \cdots - a_{1n} x_n^{(k-1)})$$

$$x_2^{(k)} = \frac{1}{a_{22}} (b_2 - a_{21} x_1^{(k)} - a_{23} x_3^{(k-1)} - \cdots - a_{2n} x_n^{(k-1)})$$

$$\vdots$$

$$x_n^{(k)} = \frac{1}{a_{nn}} (b_n - a_{n1} x_1^{(k)} - a_{n2} x_2^{(k)} - \cdots - a_{n,n-1} x_{n-1}^{(k)})$$

Define $A = D + L + U$. Starting from $Lx^{(k)} + Dx^{(k)} + Ux^{(k-1)} = b$, we get:

$(D + L)x^{(k)} = -Ux^{(k-1)} + b$

Then the Gauss-Seidel iterative algorithm can be written in matrix form as:

$x^{(k)} = -(D + L)^{-1}Ux^{(k-1)} + (D + L)^{-1}b,$ for $k \geq 1$.

Received: August 12 2025

International Journal of Applied Mathematics

Volume 38 No. 9s , 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

In this context, D represents the diagonal component of matrix A, L denotes the strictly lower triangular component of A, and U signifies the strictly upper triangular component of A. This matrix representation offers a concise formulation of the iterative Gauss- Seidel method.

3.2.2 Rate of Convergence

Corollary 6 (6)**.** *(i) implies* $\|T\| < 1$*, where* $\rho(T)$ *is the spectral radius of* T*.*

4 Fully Fuzzy Linear System

Consider the $m \times n$ linear system of equations:

$\tilde{a}_{i1}\tilde{x}_1 \oplus \tilde{a}_{i2}\tilde{x}_2 \oplus \cdots \oplus \tilde{a}_{in}\tilde{x}_n = \tilde{b}_i,$ for $i = 1, 2, \ldots, m.$

In matrix notation, this system can be expressed as:

$\tilde{A} \otimes \tilde{x} = \tilde{b}$ or simply $\tilde{A}\tilde{x} = \tilde{b}$

where: $\tilde{A}$ is an $m \times n$ fuzzy matrix with nonnegative entries, $\tilde{x}$ represents the unknown nonnegative fuzzy vector, $\tilde{b}$ denotes the known arbitrary fuzzy vector.

This system is referred to as a **nonnegative fully fuzzy linear system (FFLS)**, where the left and right spreads (y and z) are fixed.

To compute $\tilde{A} \otimes \tilde{x}$, we utilize Definition 3, resulting in the following system of equations:

$$\begin{aligned} Ax &= b \\ (A + M)y + (A - N)z &= g \\ (A - N)y + (A + M)z &= h \end{aligned}$$

Here, A represents the matrix of core values from the fuzzy matrix $\tilde{A}$, while M and N are matrices derived from the spread param- eters of $\tilde{A}$. The vectors x, y, z correspond to the core and spreadparameters of the fuzzy vector $\tilde{x}$, and b, g, h correspond to the core and spread parameters of the fuzzy vector $\tilde{b}$.

5 Application: Numerical Solution of the 4×4 Non-Symmetric Fuzzy Linear System Using the Jacobi Method

System Representation

The given 4×4 non-symmetric fuzzy linear system is:

$\tilde{A}\tilde{x} = \tilde{b}$

(Note: The specific components of the matrices $\tilde{A}$ and $\tilde{b}$ are not provided in the source text, so we use the general matrix form.)

5.1 Jacobi Method Implementation

Tolerance is set at $\epsilon = 10^{-6}$. The maximum iterations allowed is $N = 500$, and the initial guess for all variables is $\tilde{x}^{(0)} = (0, 0, 0, 0)^T$. For each $\alpha \in [0, 1]$, the system is

Received: August 12 2025

Volume 38 No. 9s , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

solved for both lower and upper bounds. For each iteration k from 1 to N, i from 1 to 4,

compute the lower bound contribution from other variables:

$$x^{(k)}(\alpha) = \frac{1}{a_{ii}(\alpha)} \left(b_i(\alpha) - \sum_{j=1, j=i} a_{ij} x_j^{(k-1)} \right)$$

Next, compute the upper bound contribution from other variables:

$$\bar{x}^{(k)}(\alpha) = \frac{1}{\bar{a}_{ii}(\alpha)} \left(\bar{b}_i(\alpha) - \sum_{j=1, j=i} a_{ij} \bar{x}^{(k-1)} \right)$$

(Note: The original text uses a_{ij} for both lower and upper bound equations, which might be a simplification or error in the source material. I've retained a_{ij} for the summation term.)

The upper bound solution: x¯(k)(α) = (x¯(k)(α), x¯(k)(α), x¯(k)(α), x¯(k)(α))T .
1 2 3 4

Finally, check for convergence: If max

|x(k)(α) − x(k−1)(α)| < ϵ

stop and return x(k).

i i i

Construct Fuzzy Solution
After convergence, the solution for each x˜j is:
x˜j = [xj(α), x¯j(α)]

5.1.1 Numerical Results

After implementing the Jacobi method with the specified parame- ters, the solution converges after **342** iterations with a final residual of $\mathbf{7.2} \times \mathbf{10^{-7}}$.
The numerical solution for the fuzzy linear system (for a specific α level, usually $\alpha = 0$ or $\alpha = 1$ depending on context, which is not specified) is:
$\tilde{x}_1 = [2.1, 2.1]$
$\tilde{x}_2 = [1.85, 1.85]$
$\tilde{x}_3 = [1.5, 1.5]$
$\tilde{x}_4 = [0.95, 0.95]$
These results represent the fuzzy numbers that satisfy the given non-symmetric fuzzy linear system within the specified tolerance.

Iteration Table (Jacobi Method)

Gauss-Seidel Method Implementation

For each equation i from 1 to 4: Compute contributions using the latest values for $x^{(k)}$ (where $j < i$) and previous values for $x^{(k-1)}$ (where $j \geq i$):

$$x^{(k)}(\alpha) = \frac{1}{} \left(b_i(\alpha) - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)}(\alpha) - \sum_{j=i} \right)$$

Received: August 12 2025

Volume 38 No. 9s , 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

#

$a\ x^{(k-1)}(\alpha)$ $_i$ $\overline{a_{ii}(\alpha)}$ $_i$ $_{ij\ j}$ $_{j=1}$ $_{j=i+1}$ ^{n} $_{ij\ j}$

Table 1: Jacobi Method Iteration Summary

Iter Residual	x_1		x_2		x_3		x_4	
	$x_1(\alpha)$	$\bar{x}_1(\alpha)$	$x_2(\alpha)$	$\bar{x}_2(\alpha)$	$x_3(\alpha)$	$\bar{x}_3(\alpha)$	$x_4(\alpha)$	$\bar{x}_4(\alpha)$
1.0 3.25	2.0	2.0	0.75	0.75	1.375	1.375	0.875	0.875
2.0 3.25	0.75	0.75	1.375	1.375	0.875	0.875	0.875	0.875
3.0 1.6	2.3594	2.3594	2.0859	2.0859	1.6562	1.6562	0.9609	0.9609
. . .	.	.	.	.	.	.	.	.
338.0 7.2e − 07	2.1	2.1	1.85	1.85	1.5	1.5	0.95	0.95

$\bar{x}^{(k)}(\alpha) =$ $_i$ $\frac{1}{\bar{a}_{ii}(\alpha)}$ $\bar{b}_i(\alpha) - \sum_{j=1}^{i-1} \bar{a}_{ij} \bar{x}_j^{(k)}(\alpha) - \sum_{j=i+1} \bar{a}_{ij}$ $_j$ $\bar{x}^{(k-1)}(\alpha)$

$\bar{x}^{(k-1)}$

#

(α)

Iteration Table (SOR Method)

5.2 SOR Method Implementation

For each equation i from 1 to 4: Compute contributions using the latest values for $x^{(k)}$ (where $j < i$) and previous values for $x^{(k-1)}$ $_j$ $_i$ (where $j \geq i$):

$\tilde{x}^{(k)}(\alpha) =$ $_i$ $\frac{1}{\overline{a_{ii}(\alpha)}}$ $b\ (\alpha) - \sum_{j=1}^{i-1} a_{ij}\ x^{(k)}(\alpha) - \sum_{j=i+1}^{n} a_{ij}\ x^{(k-1)}(\alpha)$ $_j$ #

$\bar{\tilde{x}}^{(k)}(\alpha) =$ $_i$ $\frac{1}{\bar{a}_{ii}(\alpha)}$ $\bar{b}_i(\alpha) - \sum_{j=1}^{i-1} \bar{a}_{ij} \bar{x}_j^{(k)}(\alpha) - \sum_{j=i+1}^{n} \bar{a}_{ij} \bar{x}_j^{(k-1)}(\alpha)$ #

Received: August 12 2025

Volume 38 No. 9s , 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Update $x_i^{(k)}(\alpha)$ and $\bar{x}_i^{(k)}(\alpha)$ using the relaxation factor ω:

$$x_i^{(k)}(\alpha) = x_i^{(k-1)}(\alpha) + \omega(\tilde{x}_i^{(k)}(\alpha) - x_i^{(k-1)}(\alpha))$$

$$\bar{x}_i^{(k)}(\alpha) = \bar{x}_i^{(k-1)}(\alpha) + \omega(\bar{\tilde{x}}_i^{(k)}(\alpha) - \bar{x}_i^{(k-1)}(\alpha))$$

Table 2: Gauss–Seidel Method Iteration Summary

Iter	x_1		x_2		x_3		x_4		Residual
	$\underline{x}_1(\alpha)$	$\bar{x}_1(\alpha)$	$\underline{x}_2(\alpha)$	$\bar{x}_2(\alpha)$	$\underline{x}_3(\alpha)$	$\bar{x}_3(\alpha)$	$\underline{x}_4(\alpha)$	$\bar{x}_4(\alpha)$	
1.0	4.0	4.0	2.5	2.5	9.0	9.0	3.5	3.5	9.0
2.0	1.2	1.2	1.6	1.6	1.1	1.1	0.9	0.9	4.5
3.0	2.05	2.05	1.92	1.92	1.48	1.48	0.955	0.955	1.2
.	.	.	.	.	.	.	.	.	.
.	.								
170.0	2.1	2.1	1.85	1.85	1.5	1.5	0.95	0.95	7.2e − 07

6 The Gauss-Seidel (GC) Method

The **Gauss-Seidel (GC) Method** is an iterative technique used to solve a system of linear equations. It is an improvement over the Jacobi method, as it updates the solution vector after each cal- culation. In the Gauss-Seidel method, each unknown is updated using the latest values of other unknowns as soon as they are com- puted, leading to faster convergence in many cases. The Gauss- Seidel method is used for solving a system of linear equations:

$$AX = B$$

Where: A is the matrix of coefficients. X is the vector of unknowns. B is the constant vector. The method iteratively solves for each unknown, updating them as soon as the new value is computed. The equation for updating each variable xi at iteration k + 1 is given by:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x^{(k)}\right) \quad \text{for } i = 1, 2, \ldots, n.$$

Received: August 12 2025

International Journal of Applied Mathematics

Volume 38 No. 9s , 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Table 3: SOR Method Iteration Summary

Iter / Residual	x_1		x_2		x_3		x_4	
	$\underline{x}_1(\alpha)$	$\bar{x}_1(\alpha)$	$\underline{x}_2(\alpha)$	$\bar{x}_2(\alpha)$	$\underline{x}_3(\alpha)$	$\bar{x}_3(\alpha)$	$\underline{x}_4(\alpha)$	$\bar{x}_4(\alpha)$
1.0 9.0	4.0	4.0	2.5	2.5	9.0	9.0	3.5	3.5
2.0 3.8	1.68	1.68	1.96	1.96	1.54	1.54	0.98	0.98
3.0 1.2	2.112	2.112	1.8928	1.8928	1.5048	1.5048	0.9522	0.9522
. .	.	.	.	.	.	.	.	.
93.0 $7.2e-07$	2.1	2.1	1.85	1.85	1.5	1.5	0.95	0.95

6.1 Selecting the ω Parameter using Conjugate Gradient

Consider the linear system $Ax = b$ with A symmetric positive def- inite. Let $x^{(k+1)}$ be the approximation from $x^{(k)}$ by the relaxed forward Gauss-Seidel sweep:

$$x^{(k+1)} = (D + \omega L)^{-1}[(1 - \omega)D - \omega U]x^{(k)} + \omega(D + \omega L)^{-1}b$$

When interval data are present, write for each i the lower and upper contributions $L^{(k+1)}$ and $U^{(k+1)}$. The relaxed updates become:

$$\underline{x}_i^{(k+1)}(\alpha) = \underline{x}_i^{(k)}(\alpha) + \omega(L_i^{(k+1)}(\alpha) - \underline{x}_i^{(k)}(\alpha))$$
$$\bar{x}_i^{(k+1)}(\alpha) = \bar{x}_i^{(k)}(\alpha) + \omega(U_i^{(k+1)}(\alpha) - \bar{x}_i^{(k)}(\alpha))$$

(Note: $L_i^{(k+1)}$ and $U_i^{(k+1)}$ represent the Gauss-Seidel (unrelaxed) updates for the lower and upper α-cuts, respectively, as defined in the previous section.)

The choice of ω is guided by the spectrum of the Jacobi op- erator. Define $A = D + L + U$ and the Jacobi iteration matrix $B = -D^{-1}(L+U)$. If $\lambda_{\min}$ and $\lambda_{\max}$ denote the extremal eigenvalues of B, then the Jacobi spectral radius is $\rho(B) = \max(|\lambda_{\min}|, |\lambda_{\max}|)$.

For diagonally dominant and SPD problems the near-optimal relaxation is well-approximated by **Young's expression**:

$$\omega^* = \frac{2}{1 + \sqrt{1 - \rho(B)^2}}$$

To evaluate $\rho(B)$ without forming eigenvalue decompositions, a short **Lanczos process** is applied to B, which is equivalent to **Conjugate Gradient (CG)** on the preconditioned system $D^{-1}Ax = D^{-1}b$. Let T_k be the Lanczos tridiagonal built from k steps; its Ritz eigenvalues approximate the extremal eigenvalues of B. Using $\lambda_{\min}$ and $\lambda_{\max}$ obtained from T_k, one takes $\rho(B) = \max(|\lambda_{\min}|, |\lambda_{\max}|)$, inserts it into Young's expression to obtain ω^*, and finally vali- dates the choice by observing the residual contraction of a few SOR

Received: August 12 2025

Volume 38 No. 9s , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

sweeps. This procedure is intrinsic to the nominal system and re- mains valid when interval bounds are present, since ω depends on spectral information rather than individual samples.

In the present instance a concrete SPD system is constructed as $A = MM^T + 0.1I$, M is a 4×4 random matrix (I with a fixed random seed), and b is taken as $b = A\mathbf{1}$ with $\mathbf{1} = (1, 1, 1, 1)^T$. Lanczos applied to $B = I - D^{-1}A$ yields $\lambda_{\min} \approx -0.428$ and $\lambda_{\max} \approx 0.722$, so $\rho(B) \approx 0.722$. The resulting Young estimate is $\omega^* \approx 1.411$. A brief local validation identifies an effective relaxation $\omega \approx 1.4$ for the ensuing SOR iterations.

Starting from the zero vector, the residual $\|x^{(k)} - x^*\|$ decreases monotonically under SOR with the selected relaxation. The figure shows the semilogarithmic decay of $\|r^{(k)}\|$; convergence to tolerance 10^{-6} is achieved in **9** iterations. In the non-interval case displayed
here one has $x_i^{(k)}(\alpha) = \bar{x}_i^{(k)}(\alpha)$ at every step, and this identity is
reflected in the table below.

Iteration Table (Proposed Method)

Table 4: Proposed Method Iteration

Iter	x1_low	x1_up	x2_low	x2_up	x3_low	x3_up	x4_low	x4_up
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	2.273276	2.273276	1.984474	1.984474	1.791921	1.791921	1.017642	1.017642
2	2.099921	2.099921	1.852793	1.852793	1.461288	1.461288	0.944458	0.944458
3	2.097747	2.097747	1.847150	1.847150	1.504312	1.504312	0.950701	0.950701
4	2.100401	2.100401	1.850606	1.850606	1.499552	1.499552	0.949890	0.949890
5	2.099956	2.099956	1.849903	1.849903	1.500047	1.500047	0.950016	0.950016
6	2.100003	2.100003	1.850014	1.850014	1.499995	1.499995	0.949998	0.949998
7	2.100000	2.100000	1.849998	1.849998	1.500001	1.500001	0.950000	0.950000
8	2.100000	2.100000	1.850000	1.850000	1.500000	1.500000	0.950000	0.950000
9	2.100000	2.100000	1.850000	1.850000	1.500000	1.500000	0.950000	0.950000

The method above supplies a mathematically coherent selection of the relaxation parameter directly from spectral data of the Jacobi- preconditioned operator, with the Conjugate Gradient process fur- nishing those spectral surrogates at negligible cost. When interval coefficients are present, the same selection can be performed on a nominal or midpoint matrix, after which the interval SOR up- dates propagate the lower and upper bounds according to the given $\alpha, \beta, \gamma, \delta$ parameters.

7 Conclusion

This study compared the performance of Jacobi, Gauss-Seidel, and Successive Over-Relaxation (*SOR*) methods in solving a 4×4 non- symmetric fuzzy linear system with triangular fuzzy coefficients. All three methods converged to identical fuzzy solutions, proving their robustness in handling uncertainty bounds. However, signif- icant differences in computational efficiency were observed. The Jacobi method required over 300 iterations for convergence, while Gauss-Seidel reduced iterations by 50

Received: August 12 2025

Volume 38 No. 9s , 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

References

[1] Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338–353..

[2] Dehghan, M., Hashemi, B. (2006). Iterative solution of fuzzy linear systems. Applied Mathematics and Computation, 175(1), 645–674..

[3] Friedman, M., et al. (1998). Fuzzy linear systems. Fuzzy Sets and Systems, 96(2), 201–209.

[4] Buckley, J. J., Qu, Y. (1991). Solving systems of linear fuzzy equations. Fuzzy Sets and Systems, 43(1), 33–43.

[5] Saad, Y. (2003). Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics.

[6] Young, D. M. (1971). Iterative Solution of Large Linear Sys- tems. Academic Press.

[7] Allahviranloo, T., Ahmady, N. (2010). The parametric form of fuzzy numbers and its application. Journal of Computational and Applied Mathematics, 235(5), 1214–1225.

[8] Abbasi, F., Allahviranloo, T. (2018). A new method for solv- ing fully fuzzy linear systems with trapezoidal fuzzy numbers. Soft Computing, 22(7), 2209–2223.

[9] Hanss, M. (2005). Applied Fuzzy Arithmetic: An Introduction with Engineering Applications. Springer-Verlag.

[10] Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338–353.

[11] Kaufmann, A., Gupta, M. M. (1985). Introduction to fuzzy arithmetic: Theory and applications. Van Nostrand Reinhold.

[12] Friedman, M., et al. (1998). Fuzzy linear systems. Fuzzy Sets and Systems, 96(2), 201–209.

[13] Dehghan, M., Hashemi, B. (2006). Iterative solution of fuzzy linear systems. Applied Mathematics and Computation, 175(1), 645–674.

Received: August 12 2025