International Journal of Analysis and Applications

Structured Linear Systems and Their Iterative Solutions Through Fuzzy Poisson's Equation

I. K. Youssef[1,*], Hewayda M. S. Lotfy[2]

[1]*Mathematics Department, Faculty of Science, Islamic University of Madinah, KSA*

[2]*Ain Shams University, Faculty of Science, Department of Mathematics, Cairo, Egypt*

**Corresponding author:* kaoud22@sci.asu.edu.eg

ABSTRACT. A realistic version of the modified successive overrelaxation (MSOR) with four relaxation parameters is introduced (MMSOR) with application to a representative matrix partition. The one-dimensional Poisson's equation with fuzzy boundary values is the standard source problem for our treatment (it is sufficient to introduce all the concepts in a simple form). The finite difference method with RedBlack (RB)-Labelling of the grid points is used to introduce a fuzzy algebraic system with characterized fuzzy weak solutions (corresponding to black grid points). We introduce the algorithmic structure and the implementation of MMSOR on the de-fuzzified linear system. The choice of relaxation parameters is based on the minimum Spectral Radius (SR) of the iteration matrices. A comparison with SOR (one relaxation parameter) and MSOR (two relaxation parameters) is considered, and a relation between the three methods is revealed. Assuming the same accuracy, the experimental results showed that the MMSOR runs faster than the SOR and the MSOR methods.

1. Introduction

The problem of solving large linear systems is one of the oldest classical problems that appear in many scientific and engineering applications. Iterative methods for solving linear systems began in 1823 by Gauss, Jacobi 1853, Seidel 1878, and many others. In 1950 Young introduced the fascinating successive overrelaxation (SOR) method. Due to the coefficient matrix structure, obtained from the discretization of Poisson's equation, Young introduced the modified

Received May 12, 2024

2020 Mathematics Subject Classification. 65-04, 65-11, 65M06, 68Q06, 35-04, 68U10.

Key words and phrases. fuzzy equations; linear systems; iterative methods; Poisson's equation.

https://doi.org/10.28924/2291-8639-22-2024-103

ISSN: 2291-8639

successive overrelaxation MSOR. Poisson's equation is a partial differential equation of comprehensive utility in many fields, such as theoretical physics and engineering applications. In 1998, M. Friedman et al [1] introduced the first systematic, algorithmic treatment of fuzzy linear systems. Accordingly, many publications utilize different iterative techniques for solving fuzzy linear systems. Fuzzy linear systems appear in many applications; for example, Eng Jeng H et al. in propose an image editing method based on a Laplacian operator to discretize Poisson's equation [2]. Consequently, a linear system is attained and solved by the linear solver MSOR applying the RB-Labelling. The conclusions revealed that MSOR can solve the Poisson image blending problem effectively, requiring fewer iterations and less computational time than other methods. A review of fuzzy differential equations offers a sequential study of fuzzy differential equations (FDEs) of integer and fractional orders focusing on fuzzy derivatives of fuzzy number-valued functions [3]. The structure of the linear system (Sparse, Toeplitz, Symmetric, block structure, etc.) affects the solution technique considered, especially for large linear systems that appear in the discretization of differential equations. Such sparse systems are the key source for the presence of iterative techniques such as the SOR (successive over-relaxation) and gradient approaches and their varieties. The situation is more convenient when the linear system includes some fuzzy parameters. A frequent model for solving an arbitrary $n \times n$ FLSE whose coefficient matrix is crisp, and its right-hand side contains a fuzzy number vector was introduced by Friedman et al. [1]. They offered an embedding method where the original $n \times n$ FLSE is re-established by a $2n \times 2n$ crisp linear system. Furthermore, another embedding method was introduced by Allahviranloo et al [4]. It considers the replacement of the $n \times n$ FLSE by two $n \times n$ crisp linear systems. In both stated embedding techniques, the size of the computational work is at least doubled. Solving such crisp, large linear systems is a problem consequently, the utilization of iterative techniques becomes incredibly valuable. The iterative techniques for solving fuzzy linear systems are investigated in ([5], [6], [7], [8]). In this paper, the block structure of the system that appears in the discretization of boundary value problems is considered, and this block structure is further extended due to the appearance of fuzzy parameters. We consider boundary value problems with fuzzy boundary conditions. A general model is established for solving an $n \times n$ FLSE whose coefficients are crisp with a fuzzy right-hand side and consists of two key steps. First, employing an embedding approach to the FLSE produces a $2n\times 2n$ de-fuzzified linear system. Second, applying to the resulting de-fuzzified matrix three different iterative methods, the Successive Over Relaxation (SOR), the Modified Successive Over Relaxation (MSOR), and the new Multi-Parameter SOR (MMSOR). The later iterative technique applies SOR with a new block structure, and the solution is generated for three different partitions of the de-fuzzified matrix. The paper is organized as follows: Section 2 introduces the suggested framework for applying fuzzy iterative methods. Section 3 examines the model on some numerical examples of a fuzzy boundary value problem. The discussion of the results of the numerical examples is offered in Section 4, and finally, the conclusion is presented in Section 5.

2. Material and Methods

One of the fundamental sources of the structured large linear system is the discretization of Poisson's Equation. To exemplify the performance of iterative methods, two examples are studied. The first example is a Poisson boundary value problem with two fuzzy boundary conditions, and the second is the same problem with one fuzzy boundary condition and one crisp boundary condition. The proposed technique starts by transforming the $n \times n$ FLSE into a crisp $2n \times 2n$ linear system and next applying the suggested iterative techniques. The structure of the resulting crisp $2n \times 2n$ linear system encourages us to modify the MSOR method. A few algorithms are introduced to adopt the computational work with an application on two examples.

2.1 Fuzzy Linear Systems

The basic concepts of fuzzy numbers and FLSE are given in ([1], [9 - 13]), and we reintroduce the essential material to complete our work.

Definition 1

Given an FLSE $Ax = y$ where the coefficient matrix $A = (a_{ij}), 1 \leq i, j \leq n$ is a crisp n × n matrix and $y_i \in E^1$, $1 \leq i \leq n$. A fuzzy number vector $(x_1, x_2, \ldots, x_n)^T$ given by

$$\left(\underline{x_i}(r), \overline{x_i}(r)\right), 1 \leq i \leq n, \qquad 0 \leq r \leq 1,$$

is termed a solution of the FLSE if:

$$\begin{cases} \underline{\sum_{j=1}^{n} a_{ij}\, x_j(r)} = \sum_{j=1}^{n} \underline{a_{ij}\, x_j}\,(r) = \underline{y_i}\,(r), \;\; i = 1, \ldots, n \\ \overline{\sum_{j=1}^{n} a_{ij}\, x_j(r)} = \sum_{j=1}^{n} \overline{a_{ij}\, x_j}\,(r) = \overline{y_i}\,(r), \;\; i = 1, \ldots, n \end{cases} \tag{2.1}$$

If, for some i, $a_{ij} > 0$, $1 \leq j \leq n$, then

$$\sum_{j=1}^{n} a_{ij}\, \underline{x}_j = \underline{y}_i, i = 1, \ldots, n \quad \text{and} \quad \sum_{j=1}^{n} a_{ij}\, \overline{x}_j = \overline{y}_i, i = 1, \ldots, n$$

In general, an arbitrary equation for either $\underline{y}_i$ or $\overline{y}_i$ may incorporate a linear mix of $\underline{x}_j$'s and $\overline{x}_j$'s. Thus, a crisp $2n \times 2n$ linear system with a right-hand side column,

$(\underline{y}_1, \underline{y}_2, \cdots, \underline{y}_n, -\overline{y}_1, -\overline{y}_2, \cdots, -\overline{y}_n)^T$ needs to be solved.

Definition 2

A De-fuzzified Linear System of Equations (DFLSE) $SX = Y$ of an $n \times n$ FLSE $A\,x = y$ can be formed where S is $2n \times 2n$ crisp matrix that is initialized with zeros and updated from the coefficient matrix A as follows:

$$\begin{aligned} &If\ a_{ij} \geq 0 \ \rightarrow \ s_{ij} = s_{i+n,j+n} = a_{ij} \\ &If\ \ a_{ij} < 0 \ \rightarrow \ s_{i,j+n} = s_{i+n,j} = -a_{ij} \end{aligned} \tag{2.2}$$

The matrix S can be written in block structure as $S = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_1 \end{bmatrix}$, $S = (S_{ij}) \geq 0, 1 \leq i,j \leq 2n$. The matrix S_1 is an $n \times n$ crisp matrix that includes the nonnegative elements of matrix A, and S_2 is an $n \times n$ crisp matrix that includes the absolute values of the negative elements of A where $A = S_1 - S_2$. The right-hand side column $Y = (\underline{y}_1, \underline{y}_2, \dots, \underline{y}_n, -\overline{y}_1, -\overline{y}_2, \cdots, -\overline{y}_n)^T$. Therefore, the solution vector should be $X = (\underline{x}_1, \underline{x}_2, \dots, \underline{x}_n, -\overline{x}_1, -\overline{x}_2, \cdots, -\overline{x}_n)^T$.

Definition 3

For arbitrary fuzzy numbers $u = (\underline{u}(r), \overline{u}(r))$ and $v = (\underline{v}(r), \overline{v}(r)) \in E^1$, the amount

$$D^1(u,v) = \int_0^1 \left(\left| \underline{u}(r) - \underline{v}(r) \right| + \left| \overline{u}(r) - \overline{v}(r) \right| \right) dr, \tag{2.3}$$

is the distance between u and v. The function $D^1(u,v)$ is a metric in E^1. It is indicated that (E^1, D^1) is a complete metric space. D^1 can be used to define the error between successive calculations of fuzzy solutions in iterative methods, where the goal is to maximize the likelihood of exact solutions and the estimated solutions' nearness.

Definition 4

Let $X = \left\{ \left(\underline{x_i}(r), -\overline{x}_i(r) \right), 1 \leq i \leq n \right\}$ be a set of fuzzy numbers that denotes the unique solution for the system $SX = Y$ of the $2n \times 2n$ linear system of equations. The fuzzy number vector $\mathbf{U} = \left\{ \left(\underline{u_i}(r), \overline{u}_i(r) \right), 1 \leq i \leq n \right\}$ *is* described by:

$$\begin{aligned} \underline{u_i}(r) &= \min \left\{ \left(\underline{x_i}(r), \overline{x}_i(r), \underline{x_i}(1) \right) \right\}, \\ \overline{u}_i(r) &= \max \left\{ \left(\underline{x_i}(r), \overline{x}_i(r), \underline{x_i}(1) \right) \right\} \end{aligned} \tag{2.4}$$

and known as the fuzzy solution of, $SX = Y$. For a fuzzy number x, $[x]_1 = (\underline{x}(1), \overline{x}(1))$, the utilization of $x_i(1)$ is intended to remove the likelihood of fuzzy numbers whose related triangles possess an angle greater than 90°. If $(\underline{x_i}(r), \overline{x}_i(r))$, $1 \leq i \leq n$ are all fuzzy numbers then

$$\underline{u_i}(r) = \underline{x_i}(r), \ \ \overline{u}_i(r) = \overline{x}_i(r), \ 1 \leq i \leq n, \tag{2.5}$$

and U is called a strong fuzzy solution. Otherwise, U is a weak fuzzy solution; (for more details, see [1], [13]).

2.2 Finite Difference Approximation and RB-Labelling

Consider the following one-dimensional Poisson equation:

$-u''(t) = f(t)\,,\ a \le t \le b$ with fuzzy boundary conditions, $u(a) = \alpha, u(b) = \beta.$ (2.6)

Where α and β are given triangular fuzzy numbers. The continuous domain is superimposed with a discrete grid with equally spaced grid points shown in Fig. 1. The grid spacing h is the distance between any two consecutive grid points.

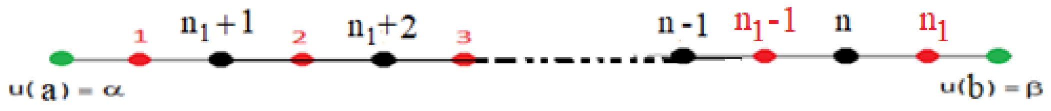


Fig. 1: RB-Labelling of a grid with n points

In the RB-Labelling, excluding the boundary points $x = a$ and $x = b$, the internal grid points are colored with minimum colors (two) such that any two consecutive points take different colors so the neighbors of any black points will be red points and vice versa. Then numbering the red points consecutively from 1 up to n_1, where n_1= *ceil (n/2)* then the black points from $(n_1 + 1)$ to n. Using the well-known central difference approximation for the second-order derivative

$$u''(t) = \frac{u(t+h)-2\,u(t)+u(t-h)}{h^2} \tag{2.7}$$

The given boundary value is equivalent to the following structured linear system (matrix equation)

$$A\,x = y, \text{ where } A = \begin{bmatrix} D_1 & B \\ B^T & D_2 \end{bmatrix}, \tag{2.8}$$

where D_1 is an $n_1 \times n_1$ diagonal matrix, D_2 is an $(n - n_1) \times (n - n_1)$ diagonal matrix and B is a rectangular matrix of order $n_1 \times (n - n_1)$, y is a column vector of dimension n with fuzzy data.

$$D_1 = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}, D_2 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}, B = \begin{bmatrix} -1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 \end{bmatrix}, |B| = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.9}$$

In the classical crisp, all red points are updated before the black points. Updating a red point only requires data from the black points, and vice versa. Therefore, the order in which points in each set are updated does not make a difference. Hence, we may think that the update is divided into two updates: a red update and a black update, and this is helpful in the convergence analysis and in the implementation development. The structure of the mentioned submatrices (2.8) will be stated explicitly in the specific case: $n = 9$, $n_1 = 5$, and are used in the numerical examples below.

2.3 The Block Structure

The associated crisp $(2n \times 2n)$ linear system is the DFLSE $S\,X = Y$

$$\mathrm{S} = \begin{bmatrix} D_1 & O & O & |B| \\ O & D_2 & |B^T| & O \\ O & |B| & D_1 & O \\ |B^T| & O & O & D_2 \end{bmatrix}. \tag{2.10}$$

The structure of the coefficient matrix **S**, suggests the use of four relaxation parameters instead of two for the original matrix A. ω_1 for the first n_1 equations (part P_1), ω_2 for the next $(n - n_1)$ equations (part P_2), ω_3 for the next $n_1$equations (part P_3) and ω_4 for the last $(n - n_1)$ equations (part P_4). We call the SOR with four relaxation parameters MMSOR. The MMSOR will coincide with the SOR if the four parameters assume the same value, and it will coincide with the MSOR if $\omega_1 = \omega_2$ and $\omega_3 = \omega_4$. The algebraic system with the above-mentioned coefficient matrices A or **S** admits a unique solution, due to the diagonal dominance and positive definiteness.

2.4 SOR, MSOR, and MMSOR Iterative Methods

The DFLSE $SX = Y$ of size $2n \times 2n$ where $\mathrm{S} \in R^{2n\times 2n}, \mathrm{Y} \in R^{2n}$ can be written in the component form as:

$$\textstyle\sum_{j=1}^{2n} s_{ij} \;\; x_j \;=\; y_i \,, s_{ii} \;\neq 0,\; i = 1, \ldots, 2n. \tag{2.11}$$

The component form of the **SOR** method is:

$$x_i^{[m+1]} = x_i^{[m]} + \frac{\omega}{s_{ii}}\left(y_i - \textstyle\sum_{j=1}^{i-1} s_{ij} \;\; x_j^{[m+1]} - \textstyle\sum_{j=i}^{2n} s_{ij} \;\; x_j^{[m]}\right),\; i = 1, \ldots, 2n \tag{2.12}$$

The efficient use of the SOR method depends on the appropriate choice of the relaxation parameter, $0 < \omega < 2$. For each problem, there is an optimal choice for ω depending on the SR of the corresponding iteration matrix. In general, one can use different values for a vector ω with length *g*. i.e., *ω (k), k = 1, …, g*. In this work, the equations corresponding to the red points are updated first.

The **MSOR** method is a variant of the SOR method, its effective use appears when the coefficient matrix can be arranged in 2×2 block form where the diagonal blocks are non-singular diagonal matrices. The matrix S can be arranged in this form, figure 3. When the equations are divided into two partitions P_1 and P_2 (red and black) then we apply a relaxation parameter for the red equations and another relaxation parameter for the black equations (see [1], [14]). We are using

ω_1(k), for the equations of P_1 and using $\omega_2\,(k)$ for P_2, where *all* $\omega's$ repeating the application over the given range, k_d=1, …, *g* and *d*=1,2. The component form updates of the MSOR method are:

$$x_i^{[m+1]} = x_i^{[m]} + \frac{\omega_1(k_1)}{s_{ii}}\Big(y_i - \textstyle\sum_{j=1}^{i-1} s_{ij}\ \ x_j^{[m+1]} - \sum_{j=i+1}^{n} s_{ij}\ \ x_j^{[m]} - s_{ii}\, x_i^{[m]}\Big), i = 1, \dots, n$$

$$x_i^{[m+1]} = x_i^{[m]} + \frac{\omega_2(k_2)}{s_{ii}}\Big(y_i - \textstyle\sum_{j=1}^{i-1} s_{ij}\ \ x_j^{[m+1]} - \sum_{j=i+1}^{2n} s_{ij}\ \ x_j^{[m]} - s_{ii}\, x_i^{[m]}\Big), i = n+1, \dots, 2n$$

$$0 < \omega_1, \omega_2 < 2\,. \qquad (2.13)$$

The **MMSOR** method is a new variant of the SOR or MSOR method which can be defined by considering four relaxation parameters for the new four partitions that appear in the S matrix. Two groups of equations are related to the red equations and the other two are related to the black equations.

MMSOR: The equations of *S* are divided into four partitions using $\omega_1(k_d)$, for the equations of P_1, $\omega_2\,(k_d)$ for P_2, $\omega_3(k_d)$ for P_3 and $\omega_4(k_d)$, for P_4, k_d=1, …, *g* and *d*=1,2,3,4 , see figure 3

The component form updates of the MMSOR method are:

$$x_i^{[m+1]} = x_i^{[m]} + \frac{\omega_{1(k_1)}}{s_{ii}}\Big(y_i - \textstyle\sum_{j=1}^{i-1} s_{ij}\ \ x_j^{[m+1]} - \sum_{j=i+1}^{2n} s_{ij}\ \ x_j^{[m]} - s_{ii}\, x_i^{[m]}\Big), i = 1, \dots, n_{1,}$$

$$x_i^{[m+1]} = x_i^{[m]} + \frac{\omega_{2(k_2)}}{s_{ii}}\Big(y_i - \textstyle\sum_{j=1}^{i-1} s_{ij}\ \ x_j^{[m+1]} - \sum_{j=i+1}^{2n} s_{ij}\ \ x_j^{[m]} - s_{ii}\, x_i^{[m]}\Big), i = n_1+1, \dots,\ n,$$

$$x_i^{[m+1]} = x_i^{[m]} + \frac{\omega_{3(k_3)}}{s_{ii}}\Big(y_i - \textstyle\sum_{j=1}^{i-1} s_{ij}\ \ x_j^{[m+1]} - \sum_{j=i+1}^{2n} s_{ij}\ \ x_j^{[m]} - s_{ii}\, x_i^{[m]}\Big), i = n+1, \dots, n+n_1$$

$$x_i^{[m+1]} = x_i^{[m]} + \frac{\omega_4(k_4)}{s_{ii}}\Big(y_i - \textstyle\sum_{j=1}^{i-1} s_{ij}\ \ x_j^{[m+1]} - \sum_{j=i+1}^{2n} s_{ij}\ \ x_j^{[m]} - s_{ii}\, x_i^{[m]}\Big), i = n+n_1+1, \dots, 2n.$$

$$(2.14)$$

If $\boldsymbol{\omega_1} = \boldsymbol{\omega_2}$, the MSOR method reduces to the SOR method, Furthermore, if $\omega_1 = \omega_2 = 1$ the MSOR method reduces to the counterpart Gauss–Seidel method. Similarly, if $\omega_1 = \omega_2$ and $\omega_3 = \omega_4$, the MMSOR method reduces to the MSOR method. From definition 2, it follows that the solution vector $X = (\underline{x}_1, \underline{x}_2, \dots, \underline{x}_n, -\overline{x}_1, -\overline{x}_2, \cdots, -\overline{x}_n)^T$ is a $2n \times 1$ solution vector. The experiments are based on the three iterative methods with the same range of values for **ω**, i.e., $\omega \in (0, 2)$ as shown in the algorithms.

2.5 Algorithmic Treatment

Given a FALSE, six self-explained algorithms are introduced to show the algorithmic treatment of the problem. In Algorithm 1, using each equation in the equation set *EQset* (*size 2n)* the function *Update* updates the solution *x* at iteration according to the assumed RB-Labelling given in Fig. 1 where the iteration m =*1, …, maxiter*. Finally, construct the solution vector *X* using the computed solution *x* by collecting the lower and upper parts of the fuzzy solution using the

RB-Labelling. The complexity for computing the defuzzification matrix is O(n^2), and the complexity is O(n^2) for each iteration in the iterative methods under consideration.

Algorithm 1 *X=RB_Update (EQset, m)*

```
Inputs: EQset is an Equation set to be updated at the iteration number (m)
Output:  the solution vector X
Procedure:
for j = 1: i-1
    check if R/B point   then Update Sol xi
for j = i+1: size (EQset)
    check if R/B point   then Update Sol xi
RB-Labelling= [1   $n_1+1$   2   $n_1+2$   3 ... $n-1$  $n_1-1$   $n$   $n_1$ ]
for i=1: n     % Construct solution vector X
    X (i, m) = x (RB-Labelling(i))            % the lower part of solution
    X (i+n, m) = x (RB-Labelling(i)+n)     % the upper part of solution
```

Algorithm 2 applies SOR (*2.12*) on each equation of *EQset* of S which is of size $2n$ and outputs the fuzzy solutions and SOR SR. This algorithm, as shown in the pseudocode, is applied on *S* using each value *ω (k), k=1, …, g* of the relaxation parameter vector. The complexity of algorithms 2, 3, and 4 is O(n^2) for each iteration for all iterative methods.

Algorithm 2 <u>**(SOR)**</u>

```
Input: Matrix S
Output: Fuzzy solution vector u, SR
Procedure: apply SOR on S given the relaxation parameter vector ω(k), k=1, …, g then:
while (1)
    for i=1 to 2n
        X= RB_Update (S, m)
    m=m+1
    perform convergence test: if ((error(m) <= tol)) or (m>=maxiter) break
u=Construct_Fuzzy_Solutions (X, Exit_iteration)
SR=Compute_SpectralRadius(S, ω)
```

Algorithm 3 applies MSOR (*2.13*) on each equation of the two sets of equations, two partitions P_1 and P_2 of S, each of size n and outputs the fuzzy solutions and MSOR SR. This algorithm, as shown in the pseudocode, is applied on the partitions P_1 and P_2 using the two relaxation parameter values $\omega_1(k_d), \omega_2(k_d)$, *where* k_d*=1, …, g and d=1,2.*

Algorithm 3 (MSOR)

```
Input: Matrix S
Output: Fuzzy solution vector u, SR
Procedure: S is divided into the partitions $P_1 = S(1: n, 1:2n)$ and $P_2 = S(n+1:2n, 1:2n)$ then apply MSOR
on $P_1$ and $P_2$ where $\omega_1(k_d), \omega_2(k_d)$, where $k_d=1, \dots, g$ and d=1,2 then:
while (1)
  for i = 1: n
              X(i)= RB_Update ($P_1$, m)
  for i = n+1: 2n
              X(i) = RB_Update ($P_2$, m)
  m=m+1
  perform convergence test: if ((error(m) <= tol)) or (m>=maxiter) break
u=Construct_Fuzzy_Solutions (X, Exit_iteration)
SR =Compute_SpectralRadius(S, ($\omega_1, \omega_2$))
```

Algorithm 4 applies MMSOR (*2.14*) on each equation of the four sets of equations either of sizes n_1 or $n - n_1$. In the previous two iterative methods and after the final iteration, the fuzzy solutions are constructed, and the MMSOR *SR* is computed. The given relaxation parameter vectors are $\omega_1, \omega_2, \omega_3, \omega_4$ for the four partitions $P_1, P_2, P_3,$ *and* P_4 of *S*. As shown in the pseudocode below, the algorithm is applied on the partitions $P_1 \dots P_4$ using the four-relaxation parameter vectors $\omega_1(k_1), \dots, \omega_4(k_4)$ where k_d*=1, …, g, d=1,…,4.*

Algorithm 4 (MMSOR)

```
Input: Matrix S
Output: Fuzzy solution vector u, SR
Procedure: S is divided into four partitions,
use $\omega_1(k_1)$, for the first partition $P_1 = S(1: n_1, 1:2n)$ rows,
use $\omega_2(k_2)$, for the second partition $P_2 = S(n_1+1: n, 1:2n)$ rows,
use $\omega_3(k_3)$, for the third partition $P_3 = S(n+1: n+n_1, 1:2n)$ rows,
use $\omega_4(k_4)$, for the fourth partition $P_4 = S\ (n+n_1+1:2n,\ 1:2n)$ rows where $k_d=1, \dots, g$, $d=1,\dots,4$, then:
while (1)
  for i = 1: $n_1$
       X(i) = RB_Update ($P_1$, m)
  for i = $n_1$+1: n
       X(i) = RB_Update ($P_2$, m)
  for i = n+1: n+$n_1$
       X(i) = RB_Update ($P_3$, m)
  for i = n+$n_1$+1:2n
       X(i) = RB_Update ($P_4$, m)
  m=m+1
  perform convergence test: if ((error(m) <= tol)) or (m>=maxiter) break
u=Construct_Fuzzy_Solutions (X, Exit_iteration)
SR=Compute_SpectralRadius (S, ($\omega_1$, $\omega_2$, $\omega_3$, $\omega_4$))
```

Algorithm 5 shows the pseudo-code of the *Construct_Fuzzy_Solutions* function that constructs the final solutions; the solutions are collected using the lower and upper parts of the calculated solutions in vector X after the exit iteration for an iterative method.

Algorithm 5 *u= Construct_Fuzzy_Solutions* (*X, Exit_iteration*)

```
Input: Solution vector X, Exit iteration
Output: Fuzzy Solution vector u= [$u_1 \dots u_{n+2}$]
Procedure
$L_i$ = Lower part of a fuzzy number,
$U_i$ = Upper part of a fuzzy number,
% $u_1$= $\alpha$, the left fuzzy boundary condition, $0 \le r \le 1$,
$u_1$= $L_1$ + $U_1$ * r
                        % The n-computed solutions, $u_2 \dots u_{n+1}$ after function Update
for i=2: n+1
    $L_i$ =obtain the straight-line equation given the variables (r, X (i-1, Exit_iteration))
    $U_i$= obtain the straight-line equation given the variables (r, -X (i-1+n, Exit_iteration))
    $u_i$ = $L_i$ + $U_i$ *r
end
% $u_{n+2}$= $\beta$, the right fuzzy boundary condition
$u_{n+2}$ = $L_{n+2}$ + $U_{n+2}$ * r
```

Algorithm 6 presents the *Compute_SpectralRadius* function pseudo code that computes the *SR* of the suggested iterative methods using the iteration matrices.

Algorithm 6 *Compute_SpectralRadius(S, ω)*

```
Input: Matrix S, vector ω of g-values
Output: Convergence_factor, ω*
Procedure:
Step 1: Case SOR:
        Let S =L+D+U
        Iteration matrix T= $((D- \omega\times L)^{-1}) \times (((1- \omega) \times D) + \omega\times U)$
        rho =max(abs(eigen_vectors(T)))
        Convergence_factor=min(rho)
        ω*= the optimal ω with minimum convergence factor
Step 2: Case MSOR:
        $D_1$= S (1: n,1: n)
        $D_2$= S (n+1: 2n, n+1: 2n)
        H= S (1: n, n+1: 2n)
        K= S (n+1:2n, 1: n)
        F= $-D_1^{-1} \times H$
        G= $-D_2^{-1} \times K$
        Iteration matrix T =$[(1-\omega_1) \times I_1 , \omega_1\times F ; (1-\omega_1) \times\omega_2\times G , \omega_1\times\omega_2\times G\times F+(1-\omega_2) \times I_2 ]$
        rho= max (abs (eigen_vectors (T)))
        Convergence_factor=min(rho)
        $(\omega_1^*, \omega_2^*)$ = the pair with minimum convergence factor
Step 3: Case MMSOR:
        using the pair $(\omega_1^*, \omega_2^*)$ from Step 2 then Compute convergence_factor using same procedure
        in step 2 replacing $(\omega_1, \omega_2)$ with $(\omega_3, \omega_4)$ to calculate the pair $(\omega_3^*, \omega_4^*)$ we get:
            $(\omega_1^*, \omega_2^*, \omega_3^*, \omega_4^*)$ = the quadruple with minimum convergence factor
```

3. Numerical Examples

We consider Poisson's equation with Dirichlet fuzzy boundary conditions. Three cases can be considered in the first case two fuzzy boundary conditions example (1), in the second case one fuzzy boundary condition at the left boundary and the other condition is crisp boundary condition example (2), and in the third case a crisp boundary condition at left boundary and a fuzzy boundary condition at other end the results are the same as the second case. We consider the finite difference method with RB-Labelling of the grid points described in the finite difference section with $n = 9$ and $n_1 = 5$. So, a corresponding fuzzy linear system $FLSE\ AU = b$ with block structure is obtained. Due to the grid ordering the fuzzy values appear at different positions in the right-hand side vector of the linear system. Then applying the embedding techniques presented by ([1], [13]) to obtain the DFLSE with coefficient matrix S, $SX = Y$.

In solving the DFLSE $SX = Y$, first the SOR and the MSOR techniques are used, to determine the suitable values of the relaxation parameters ω , ω_1 and ω_2 ([14], [15]), according to the behavior of SR of their iteration matrices. After some experiments, the relaxation parameter ω varies in the range of 1.52:0.002:1.555 with 18 possible values of **ω** in the case of SOR, 324 pairs of ω-values in the case of MSOR, and with 104976 quadruples of $\omega_1 \ldots \omega_4$- values in the n case of MMSOR. In the calculation process, a zero value is chosen as an initial guess, a tolerance is 10^{-4}, and a maximum number of 100 iterations is used as a stopping criterion in all considered iterative techniques.

Example (1): Consider the following Poisson equation:

$-u''(t) = 1.5\,\pi^2 \sin(\pi\, t)\,,\ 2 \le t \le 3$ with fuzzy boundary conditions:

$u(2.0) = \alpha = (1.2, 1.5, 1.8) = (1.2 + 0.3\,r\,, 1.8 - 0.3\,r)$,

$u(3.0) = \beta = (1.2, 1.5, 1.8) = (1.2 + 0.3\,r\,, 1.8 - 0.3\,r)$.

Following the RB-Labelling and Fig. 1 with $n = 9$ internal grid points are selected and shown in Fig. 2 there are $n_1 = 5$ red points and $n - n_1 = 4$ black points. The solutions $u_1 \ldots u_9$ are corresponding to these points.

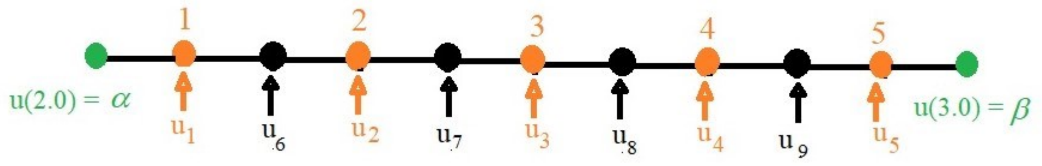
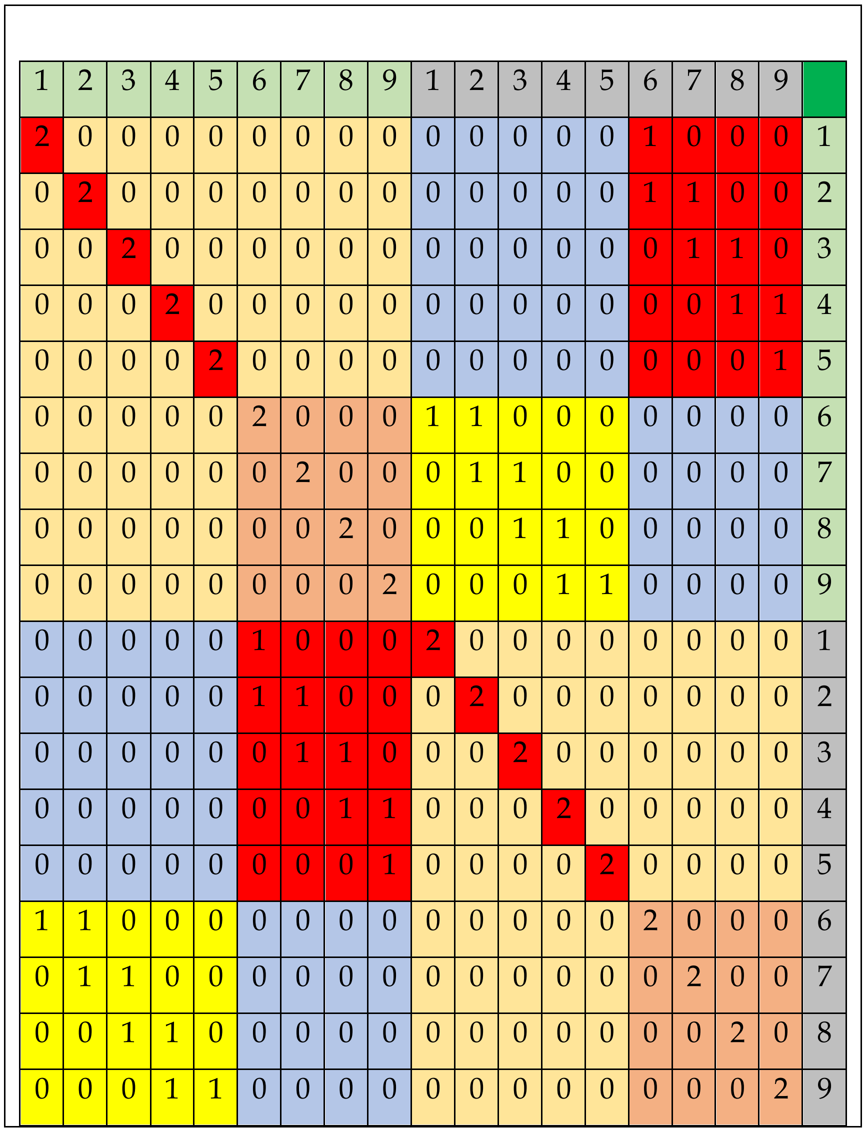


Fig. 2: RB-Labelling of the n=9 points and corresponding Fuzzy solutions

Thus, a fuzzy linear system $AX = Y is$ obtained with block-structured matrix A of size 9 $\times$ 9.

$$A = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & -1 \\ -1 & -1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 2 \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ 0.119770 \\ 0.148044 \\ 0.11977 \\ y_2 \\ 0.0870181 \\ 0.140798 \\ 0.140798 \\ 0.0870181 \end{bmatrix}, \tag{3.1}$$

where $y_1 = y_2 = (1.24575 + 0.3r, 1.84575 - 0.3r)$ are two fuzzy numbers. The resulting DFLSE S matrix for the two examples is the same and has the block structure shown in Fig. 3.



1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	2
0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	3
0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	4
0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	5
0	0	0	0	0	2	0	0	0	1	1	0	0	0	0	0	0	0	6
0	0	0	0	0	0	2	0	0	0	1	1	0	0	0	0	0	0	7
0	0	0	0	0	0	0	2	0	0	0	1	1	0	0	0	0	0	8
0	0	0	0	0	0	0	0	2	0	0	0	1	1	0	0	0	0	9
0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	1	0	0	0	2	0	0	0	0	0	0	0	2
0	0	0	0	0	0	1	1	0	0	0	2	0	0	0	0	0	0	3
0	0	0	0	0	0	0	1	1	0	0	0	2	0	0	0	0	0	4
0	0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0	5
1	1	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	6
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	7
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	2	0	8
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	2	9

Fig. 3: The Block Structure of the S matrix

The exact solution is $u(t,r) = (1.2 + 1.5\sin(\pi\, t) + 0.3\, r, 1.8 + 1.5\sin(\pi\, t) - 0.3\, r)$. The fuzzy solutions of Example 1 obtained from the three methods are the same and are shown accompanied by the exact solution in Fig 4 stated as follows:

$u_1 = (1.6676 + 0.3\, r\, , 2.2676\ - 0.3\, r)$, $u_6 = (2.6895 - 0.3001\, r, 2.0892 + 0.3\, r)$

$u_2 = (2.4241 + 0.3001\, r\, , 3.0241 - 0.3001\, r)$, $u_7 = (3.2391 - 0.3001\, r, 2.6388 + 0.3001 r)$

$u_3 = (2.7129 + 0.3001\, r\, , 3.3130 - 0.3001\, r)$, $u_8 = (3.2391 - 0.3001\, r, 2.6388 + 0.3001\, r)$

$u_4 = (2.4241 + 0.3001\, r\, , 3.0241 - 0.3001\, r)$, $u_9 = (3.0241 - 0.3001\, r, 2.4241 + 0.3001\, r\,)$,

$u_5 = (1.6676 + 0.3\, r\, , 2.2676\ - 0.3\, r)$.

The solutions' numbering matches the numbering of the grid points. It is interesting to note that the solutions corresponding to the black points are weak fuzzy solutions. It is worth noticing that in ([8], [13]), when natural labeling was used, weak solutions were obtained with even grid points.

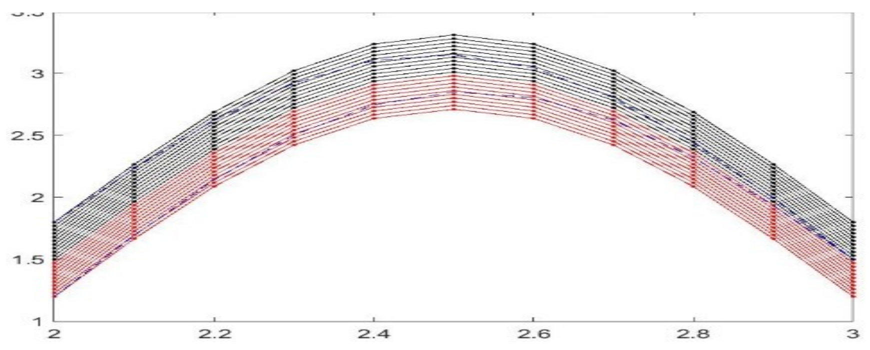


Fig. 4 Fuzzy solution of Example 1 at different alpha cuts (r =0 (0.1)1)

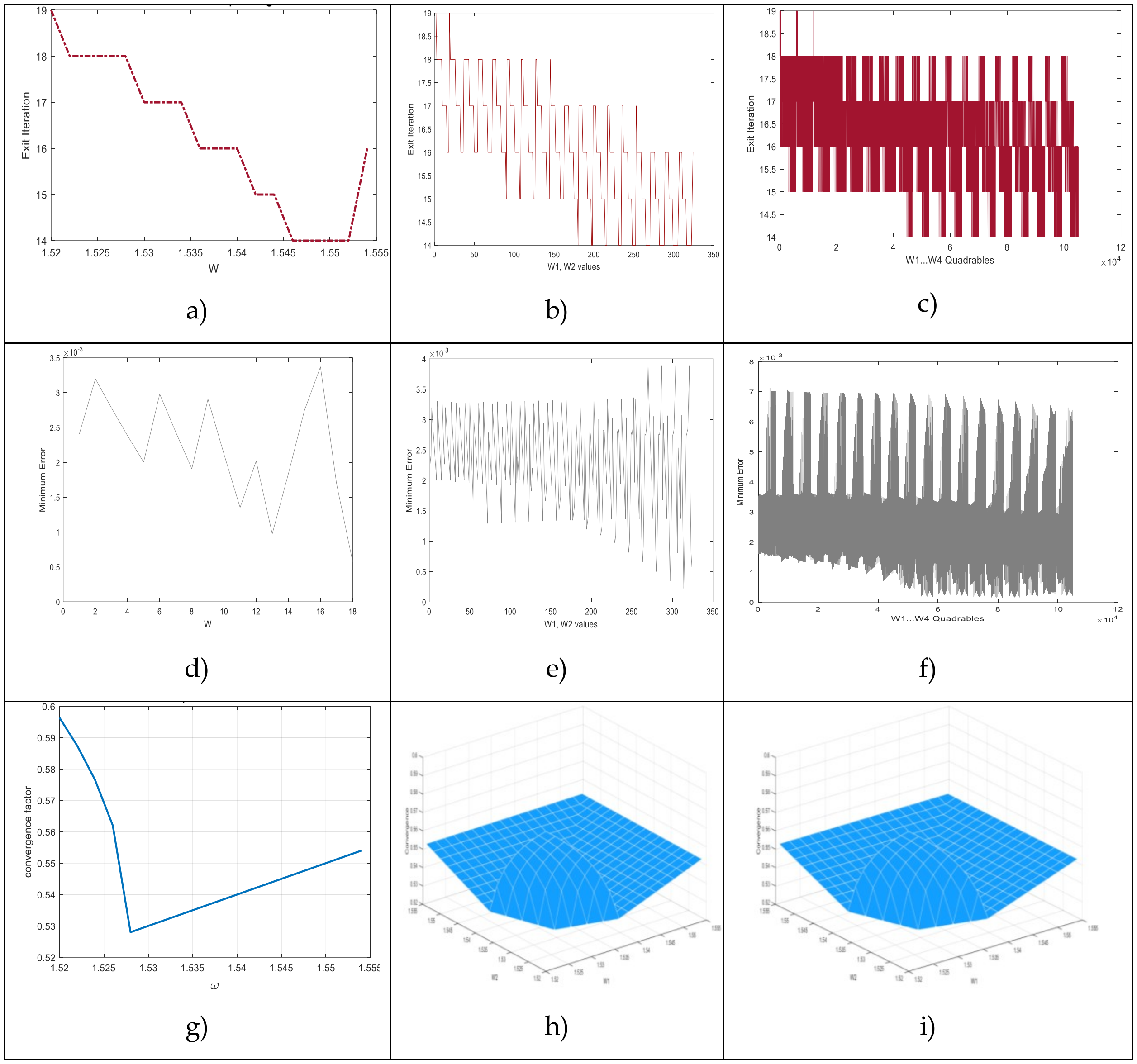


Fig. 5: SOR, MSOR, MMSOR (Example1)

Fig. 5 shows the results of Example 1. Fig. 5(a) up to Fig. 5(c) show the exit iteration profiles for applying SOR, MSOR, and MMSOR, respectively. Fig. 5(d) up to Fig. 5(f) show the minimum integral error (cap D to the 1 n Definition 3) profiles for applying SOR, MSOR, and MMSOR, respectively. The *x*-axis in Fig. 5(b), Fig. 5(c), Fig. 5(e), and Fig. 5(f) is the number of the pair/ quadruple. The *SR* is presented in Fig. 5(g) up to Fig. 5(i), the optimal quadruple $(\omega_1^*, \omega_2^*, \omega_3^*, \omega_4^*)$ = (1.258, 1.528, 1.258, 1.528) determined by MMSOR includes the optimal pair (ω_1^*, ω_2^*) = (1.258, 1.528) determined by MSOR where ω^*= 1. 258 determined by SOR. The convergence value is 0.528 for all methods.

Example (2):

Consider the same fuzzy Poisson equation: $-u''(t) = 1.5\,\pi^2 \sin(\pi\, t)\ ,\ \ 2 \le t\ \le 3$ with one fuzzy boundary condition and one crisp condition:

$$u(2.0) = \alpha\ = (1.2, 1.5, 1.8) = \ (1.2 + 0.3\,r\,, 1.8 - 0.3\,r),$$

$$u(3.0) = \beta = \ \ 1.5$$

In this case, the exact solution is:

$$u(t,r) = (0.6 + 0.3\,t + 0.9\,r - 0.3\,r\,t + 1.5\sin(\pi\,t)\,, 2.4 - 0.9\,r - 0.3t + 0.3\,r\,t + 1.5\sin(\pi\,t)).$$

A similar matrix A (*3.1*) as in Example 1 is obtained except that $y_2 = 1.545751$ in the right-hand side is a crisp value as shown in Fig. 6(b). Therefore, the S matrix is the same as in Example 1, and therefore, the exit iterations, *SR*, and ω^* are similar to those obtained in Example 1. (Fig. 5) and Fig. 7 shows the minimum error profiles of applying the SOR, MSOR, and MMSOR respectively. The obtained fuzzy solutions are:

$u_1 = (1.6976 + 0.27\,r\,, 2.2376 - 0.27\,r)$, $u_6 = (2.6295 - 0.24\,r,\ 2.1493 + 0.24\,r)$

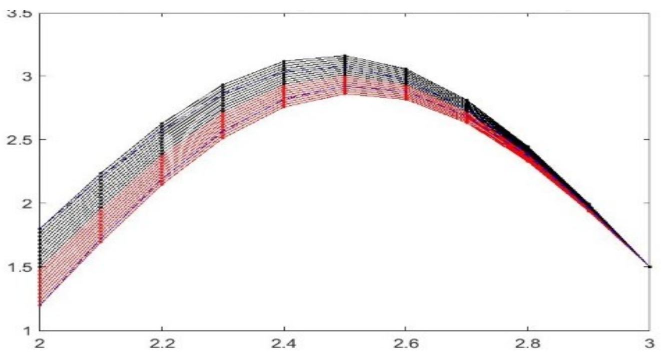
$u_2 = (2.5141 + 0.21r\,, 2.9341 - 0.21\,r)$, $u_7 = (3.1191 - 0.18\,r,\ 2.7589 + 0.18\,r)$

$u_3 = (2.863 + 0.15\,r, 3.163 - 0.15\,r)$, $\quad u_8 = (3.0591 - 0.12\,r\,, 2.8188 + 0.12\ \,r)$

$u_4 = (2.6341 + 0.09\,r\,, 2.8141 - 0.09\,r)$, $u_9 = (\ 2.4495 - 0.06\,r\,, 2.3293 + 0.06\,r)$,

$u_5 = (1.9376 + 0.03\,r\,, 1.9976\ - 0.3\,r)$.

Again, the labeling matches the grid points labeling, and the solutions corresponding to the black points are weak fuzzy solutions.



(a)

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ 0.119770 \\ 0.148044 \\ 0.11977 \\ 1.545751.5 \\ 0.0870181 \\ 0.140798 \\ 0.140798 \\ 0.0870181 \end{bmatrix}$$

(b)

Fig. 6 The Fuzzy solution of Example 2 with the exact solution at all alpha cuts

4. Results and Discussion

A 1D-Poisson's problem with fuzzy boundary conditions is considered and solved with SOR variants methods. Tables 1 and 2 show the values of ω^*, minimum iterations (*minIter)*, *SR*, and minimum error *(minError*)for each of Examples 1 and 2, respectively. The parameter ω for all methods varies within the same range from 1.52 to 1.555 with the same step size 0.002. In this way, we can observe how the choice of ω affects the iteration in all the experiments. The ω^* for SOR that gives the minimum convergence value is calculated according to [14], that is $\omega^* = 1.528$ obtained at iteration number 18 and *minError* =0.002 for all the iterative methods under investigation.

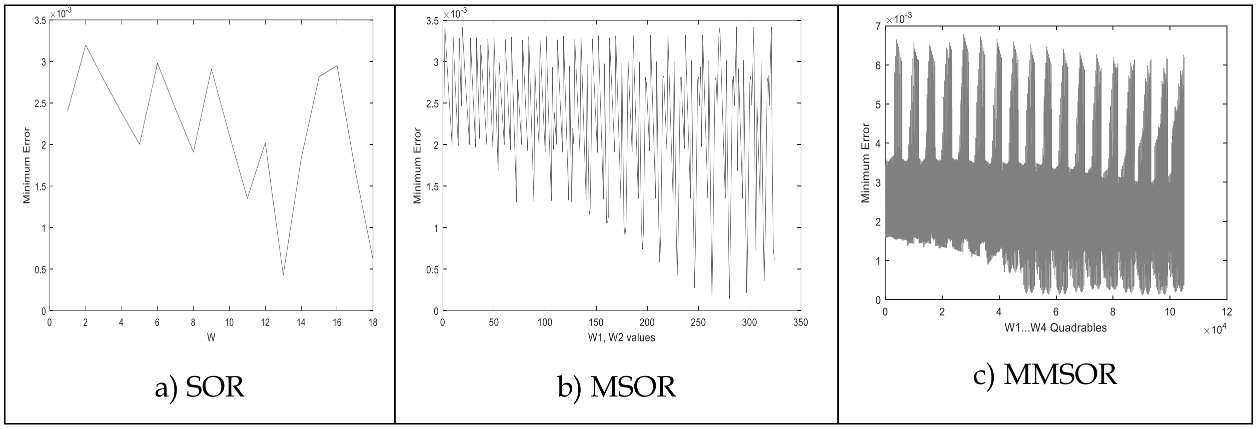


Fig. 7 SOR, MSOR, MMSOR Minimum Error profiles (Example2)

In Table 1, and similarly in Table 2. the smallest number of iterations is obtained when ω around 1.5. The optimal ω*=1.528 in SOR then this value re-appears as optimal pair (ω_1^*, ω_2^*) = (1.528, 1.528) in MSOR and as optimal quadruple $(\omega_1^*, \omega_2^*, \omega_3^*, \omega_4^*)$ = (1.528, 1.528, 1.528, 1.528) in MMSOR. In SOR, there are four ω-instances $\omega_{minIterSOR}$= (1.546, 1.548, 1.55, 1.552) that are not optimal but still gave a minimum iteration 14. These four instances in $\omega_{minIterSOR}$ re-appeared again as ω-instances pairs in MSOR, i.e., $\omega_{minIterMSOR}$ = (…, (1.546 1.548), (1.546 1.55), (1.546 1.552), (1.548 1.55), (1.548 1.552), (1.55 1.552), …) with *minimum-iteration*=14 in MSOR. These $\omega_{minIterMSOR}$ pairs re-appeared again ω-instances quadruples, i.e., $\omega_{minIterMMSOR}$ =(…, (1.546 1.548 1.546 1.55), (1.546 1.548 1.546, 1.552), (1.546 1.548 1.548 1.55), (1.546 1.548 1.55 1.552), (1.546 1.548 1.548 1.552), (1.546 1.55 1.546 1.552), (1.546 1.55 1.548 1.55), (1.546 1.55 1.548 1.552), (1.546 1.55 1.55 1.552), (1.546 1.552 1.548 1.55), (1.546 1.552 1.548 1.552), (1.546 1.552 1.55 1.552), (1.548 1.55 1.548 1.552), (1.548 1.55 1.55 1.552), (1.548 1.552 1.55 1.552), …) with minimum-iteration 14 in MMSOR. The number of their re-appearances can be calculated by combinations. In SOR, there exist 4 minimum-iteration ω-instances re-appeared as 6 pairs in MSOR (4 choose 2= 6). In MSOR, it was 6 minimum instances then re-appeared as 15 quadruples in MMSOR (6 choose 2= 15). This is a new relation between the three methods which is expressed in the following equations where 2 refers to two pairs:

Number of reappeared minimum iterations $_{\text{MSOR}} = c_2^{|\omega_{minIterSOR}|}$

Number of reappeared minimum iterations $_{\text{MMSOR}} = c_2^{|\omega_{minIterMSOR}|}$

In addition, it is worth noting that a one ω-instance $\omega_{minError}$= (1.554) with minimum error re-appeared for SOR, MSOR, and MMSOR in a singleton, in a pair, or a quadrable. The larger range of ω-values the larger range of minimum iteration values, for example, for the range (ω=1.5:0.001:1.6), the iteration range becomes 14-21 instead of 14:19 for the range (ω =1.52:0.002:1.555) and therefore the more instances of ω (ω_{minIter}) with minimum iterations *minIter*. In Table 1 and Table 2, each different run (fixing ω) for an iterative method gives fixed results. Using multiple partitions allowed the use of a range of values for up to four relaxation parameters and this allowed some pairs and quadruples to execute in less time than others. The approximate elapsed time at ω^* for SOR is 0.000002, MSOR is 0.000003, and MMSOR is 0.000001 in both examples. Hence the minimum elapsed time reached for ω^* is associated with MMSOR. In both examples, the iteration $iter_{\omega^*}$ and minimum error $minError_{\omega^*}$ for the optimal value ω^* is similar in the case of SOR, MSOR, and MMSOR. The minimum error (*minError*) reached among the three methods in the case of MMSOR is 0.000151 in Example 1 and 0.000119 in Example 2.

Table 1 and 2 Abbreviations:

minError: the minimum error reached.

$minError_{\omega^*}$ is the minimum error at ω^*.

$\omega_{minError}$ is the group of relaxation parameters associated with minimum error.

$\omega_{minIter}$ is the group of relaxation parameters associated with minimum iteration for specific iterative method.

$iter_{\omega^*}$ is the exit iteration at ω^*.

R is the exit iteration range for the relaxation parameters.

Table 1: ω^*, minimum iterations, *SR*, and minimum errors for Example 1

Example 1	ω^*	$iter_{\omega^*}$ / R & $\omega_{minIter}$	$minError_{\omega^*}$ / $minError$ & $\omega_{minError}$	***SR***
SOR • 18 - single values • Out of them 4 **minimum** iterations	1.528	**18** *R*= 14:19 where *minIter*=14 $\omega_{minIterSOR}$= (1.546, 1.548, 1.55, 1.552) at min iteration 14	$\boldsymbol{minError_{\omega^*}}$**=0.002** *minError* =0.000577 & $\omega_{minError}$= (1.554)	0.528
MSOR • 324 - pairs • Out of them 42- minimum iteration instances	(1.528, 1.528)	**18** *R*= 14:19 where *minIter*=14 $\omega_{minIterMSOR}$ = (…, (1.546 1.548), (1.546 1.55), (1.546 1.552), (1.548 1.55), (1.548 1.552), (1.55 1.552), …) 6 combinations of $\omega_{minIterSOR}$ at min iteration 14	$\boldsymbol{minError_{\omega^*}}$**=0.002** *minError* =0.000223 & $\omega_{minError}$= (1.554 1.534)	0.528
MMSOR • 104976- quadruples • Out of them 2535 minimum iteration instances	(1.528, 1.528, 1.528, 1.528)	**18** *R*= 14:19 where *minIter*=14 $\omega_{minIterMMSOR}$ =(…, (1.546 1.548 1.546 1.55), (1.546 1.548 1.546, 1.552), (1.546 1.548 1.548 1.55), (1.546 1.548 1.55 1.552), (1.546 1.548 1.548 1.552), (1.546 1.55 1.546 1.552), (1.546 1.55 1.548 1.55), (1.546 1.55 1.548 1.552), (1.546 1.55 1.55 1.552), (1.546 1.552 1.548 1.55), (1.546 1.552 1.548 1.552), (1.546 1.552 1.55 1.552), (1.548 1.55 1.548 1.552), (1.548 1.55 1.55 1.552), (1.548 1.552 1.55 1.552),…) 15 combinations of $\omega_{minIterSOR}$ at min iteration 14	$\boldsymbol{minError_{\omega^*}}$**=0.002** ***minError***=0.000151 & $\omega_{minError}$= (1.546 1.554 1.534 1.544)	0.528

Table 2: ω*, minimum iterations, *SR* and minimum errors for Example 2

Example2	ω^*	***iter***$_{\omega^*}$ / ***R & ω***$_{minIter}$	***minError***$_{\omega^*}$ / *minError &* $\omega_{minError}$	***SR***
SOR • 18 - values • 4 minimum iterations	1.528	**18** *R*=14:19 where *minIter*=14 $\omega_{minIterSOR}$ =(1.546, 1.548, 1.55, 1.552) at min iteration 14	***minError***$_{\omega^*}$**=0.002** *minError* =0.000429 & $\omega_{minError}$ =1.544	0.528
MSOR • 324 - pairs • 42- minimum iteration instances	(1.528 1.528)	**18** *R*=14:19 where *minIter*=14 $\omega_{minIterMSOR}$= (…, (1.546 1.548), (1.546 1.55), (1.546 1.552), (1.548 1.55), (1.548 1.552), (1.55 1.552),….) the 6 combinations for $\omega_{minIterSOR}$ at min iteration 14	***minError***$_{\omega^*}$**=0.002** *minError* =0.000147 & $\omega_{minError}$ = One pair with minimum error (1.55 1.538)	0.528
MMSOR • 104976- quadruples • 2545 minimum iteration instances	(1.528, 1.528, 1.528, 1.528)	**18** *R*=14:19 where *minIter*=14 $\omega_{minIterMMSOR}$ = =(…, (1.546 1.548 1.546 1.55), (1.546 1.548 1.546, 1.552), (1.546 1.548 1.548 1.55), (1.546 1.548 1.55 1.552), (1.546 1.548 1.548 1.552), (1.546 1.55 1.546 1.552), (1.546 1.55 1.548 1.55), (1.546 1.55 1.548 1.552), (1.546 1.55 1.55 1.552), (1.546 1.552 1.548 1.55), (1.546 1.552 1.548 1.552), (1.546 1.552 1.55 1.552), (1.548 1.55 1.548 1.552), (1.548 1.55 1.55 1.552), (1.548 1.552 1.55 1.552) …) the 15 combinations for $\omega_{minIterMSOR}$ at min iteration 14	***minError***$_{\omega^*}$**=0.002** ***minError =0.000119*** & $\omega_{minError}$ =(1.55 1.536 1.55 1.538) a quadruple with minimum error	0.528

5. Conclusion

In this work, we introduced a 1D Poisson with a fuzzy boundary condition, which is discretized using the finite difference method and the RB-Labelling producing a fuzzy linear system of equations. A variant of SOR, the iterative method MMSOR is presented and is applied on the DFLSE. The method is compared with the well-known iterative methods SOR and MSOR. The MMSOR uses four relaxation parameters which are applied on four partitions of S. The algebraic structure corresponding to these values is given. The experimental results showed that the same optimal relaxation parameter ω^* re-appeared in all methods. In addition, the relaxation parameter instances associated with the minimum iteration reappeared in all the methods as a pair or quadruple. It is revealed that MMSOR runs faster compared with SOR and MSOR. We look forward to considering the effect of refinement techniques [16] and the role of the relaxation parameters [17].

Conflicts of Interest: The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] M. Friedman, M. Ming, A. Kandel, Fuzzy Linear Systems, Fuzzy Sets Syst. 96 (1998), 201–209. https://doi.org/10.1016/s0165-0114(96)00270-9.

[2] E.J. Hong, A. Saudi, J. Sulaiman, Numerical Assessment for Poisson Image Blending Problem Using MSOR Iteration via Five-Point Laplacian Operator, J. Phys.: Conf. Ser. 890 (2017), 012010. https://doi.org/10.1088/1742-6596/890/1/012010.

[3] M. Mazandarani, L. Xiu, A Review on Fuzzy Differential Equations, IEEE Access 9 (2021), 62195–62211. https://doi.org/10.1109/access.2021.3074245.

[4] T. Allahviranloo, A. Hashemi, The Embedding Method to Obtain the Solution of Fuzzy Linear Systems, Int. J. Ind. Math. 6 (2014), 229-233.

[5] T. Allahviranloo, Successive Over Relaxation Iterative Method for Fuzzy System of Linear Equations, Appl. Math. Comput. 162 (2005), 189–196. https://doi.org/10.1016/j.amc.2003.12.085.

[6] Y. Feng, An Iterative Method for Fuzzy Linear Systems, in: 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, IEEE, Jinan Shandong, China, 2008: pp. 565–569. https://doi.org/10.1109/FSKD.2008.92.

[7] J.F. Yin, K. Wang, Splitting Iterative Methods for Fuzzy System of Linear Equations, Comput. Math. Model. 20 (2009), 326–335. https://doi.org/10.1007/s10598-009-9039-9.

[8] F. Abbasi, T. Allahviranloo, Solving Fully Fuzzy Linear System: A New Solution Concept, Inf. Sci. 589 (2022), 608–635. https://doi.org/10.1016/j.ins.2022.01.004.

[9] R. Goetschel Jr., W. Voxman, Elementary Fuzzy Calculus, Fuzzy Sets Syst. 18 (1986), 31–43. https://doi.org/10.1016/0165-0114(86)90026-6.

[10] T. Allahviranloo, Numerical Methods for Fuzzy System of Linear Equations, Appl. Math. Comput. 155 (2004), 493–502. https://doi.org/10.1016/s0096-3003(03)00793-8.

[11] M. Dehghan, B. Hashemi, Iterative Solution of Fuzzy Linear Systems, Appl. Math. Comput. 175 (2006), 645–674. https://doi.org/10.1016/j.amc.2005.07.033.

[12] N. Gasilov, Ş.E. Amrahov, A.G. Fatullayev, Solution of Linear Differential Equations with Fuzzy Boundary Values, Fuzzy Sets Syst. 257 (2014), 169–183. https://doi.org/10.1016/j.fss.2013.08.008.

[13] H.M.S. Lotfy, A.A. Taha, I.K. Youssef, Fuzzy Linear Systems via Boundary Value Problem, Soft Comput. 23 (2018), 9647–9655. https://doi.org/10.1007/s00500-018-3529-7.

[14] I.K. Youssef, A.A. Taha, On the Modified Successive Overrelaxation Method, Appl. Math. Comput. 219 (2013), 4601–4613. https://doi.org/10.1016/j.amc.2012.10.071.

[15] D.R. Kincaid, D.M. Young, The Modified Successive Overrelaxation Method with Fixed Parameters, Math. Comput. 26 (1972), 705–717. https://doi.org/10.1090/s0025-5718-1972-0331746-2.

[16] S.A. Meligy, I.K. Youssef, A Refinement of the KSOR Iterative Method, Int. J. Math. Comput. Sci. 17 (2022), 1193-11199.

[17] S.A. Meligy, I.K. Youssef, Relaxation Parameters and Composite Refinement Techniques, Results Appl. Math. 15 (2022), 100282. https://doi.org/10.1016/j.rinam.2022.100282.