SOR Iterative Method with Simpson’s 1/3 Rule for the Numerical Solution of Fuzzy Second Kind Fredholm Integral Equations

L H Ali, J Sulaiman and S R M Hashim

1,2,3Mathematics with Economics Programme, Faculty of Science and Natural Resources, Universiti Malaysia Sabah, 88400 Kota Kinabalu, Sabah, Malaysia.

labiyanal5@gmail.com, jumat@ums.edu.my and rahayu@ums.edu.my

Abstract. In this study, we present the application of Successive Over-Relaxation (SOR) iterative method to solve fuzzy Fredholm integral equations of the second kind (FFIE-2). In addition to that, the Simpson’s 1/3 quadrature rule is applied to derive the approximate solution of FFIE-2. Then, we use the approximate equation to generate a system of linear equations. Next, SOR iterative method is introduced to solve the generated system of linear equations. Moreover, we conduct some numerical examples to illustrate the applicability of the SOR iterative method. Finally, we discuss the efficiency of the proposed method by comparing the number of iterations, computational time and Hausdorff distance. Based on the numerical results, we conclude that SOR method is better than Jacobi and Gauss-Seidel iterative methods.

1. Introduction

Nowadays, the fuzzy integral equation is one of the important topics among researchers. Fuzzy integral equations is a unique topic which can be applied in various research fields such as artificial intelligence and robotics, image processing, biological and medical science, applied operations research, economics, geographic, quantum optics and gravity, sociology and psychology [1]. Therefore, many researchers have been studied the methods to solve fuzzy Fredholm integral equations (FFIE). In [2], Barkhordary et al. proposed trapezoidal rule for solving FFIE. Meanwhile, some researchers have introduced arbitrary kernels [3], Legendre polynomials [4], direct method using triangular functions [5], hybrid of block-pulse functions and Taylor series [6] to obtain the numerical solution of FFIE-2. Besides that, some researchers also proposed Homotopy analysis method (HAM) [7], Nystrom method [8] and Adomian decomposition method (ADM) [9] to solve linear FFIE-2. Also, Fadraji et al. [10] presented artificial neural networks method to find the solution of linear FFIE-2. In [1], Shamivand et al. studied the solution of FFIE with the degenerate kernel. Besides that, Ezzati and Ziari [11] proposed numerical solution of nonlinear FFIE by using iterative method. Meanwhile, Otadi and Mosleh [12] presented numerical solution of nonlinear FFIE by using Newton-Cotes (NC) method.

Recently, Gholam and Ezzati [13] introduced iterative method and Simpson quadrature rule to solve linear FFIE-2. Therefore, we study the application of Successive Over-Relaxation (SOR) iterative method with Simpson’s 1/3 rule for solving linear FFIE-2. Generally, Fredholm integral equation of the second kind is [3]

$$x(s) = f(s) + \lambda \int_a^b k(s,t)x(t)dt$$

(1.1)
where $\lambda > 0$, $k(s,t)$ is an arbitrary kernel function over the square $a \leq s,t \leq b$ and $f(s)$ is a function of $s : a \leq s \leq b$. The solution of problem (1.1) is a crisp if $f(s)$ is a crisp function. However, if $f(s)$ is a fuzzy function, equation (1.1) may also possess fuzzy function. Therefore we have [1,3]

$$\tilde{x}(s,r) = \bar{f}(s,r) + \lambda \int_a^b k(s,t)\tilde{x}(t)dt \quad (1.2)$$

where $\tilde{x}(s,r) = (\tilde{x}(s,r), \bar{x}(s,r))$ and $\bar{f}(s,r) = (\bar{f}(s,r), \bar{f}(s,r))$ with $0 \leq r \leq 1$.

The paper is structured in following way: in Section 2, we briefly introduced the most basic notation used in the fuzzy calculus. In Section 3, we introduce the discretization scheme based on Simpson’s 1/3 approximate equations and discuss the derivation of Simpson’s 1/3 approximate equation of linear FFIE-2. Then, we form the generated system of linear equations. In Section 4, we present the proposed method to solve linear FFIE-2. In Section 5, we introduce some numerical examples for the problem of linear FFIE-2. Finally, we present the numerical results and discuss the performances of the Jacobi, Gauss-Seidel (GS) and SOR iterative methods.

2. Preliminaries

In this section, the basic notation used in the fuzzy calculus is introduced in the several definitions. The fuzzy number is defined as follows [1,2,13,14]

2.1. Definition

A fuzzy number is a fuzzy set $u : \mathbb{R}^1 \rightarrow I = [0,1]$ which satisfies the following requirements

i. $u$ is upper semi continuous,

ii. $u(\bar{x}) = 0$ outside some interval $[a,d]$, 

iii. There are real numbers $b,c : a \leq b \leq c \leq d$ for which:

a. $u(x)$ is monotonically increasing on $[a,b]$,

b. $u(x)$ is monotonically decreasing on $[c,d]$.

c. $u(x) = 1, b \leq x \leq c$. The set of all fuzzy numbers is denoted by $E^1$.

2.2. Definition

A fuzzy number $u$ is a pair $(u, \tilde{u})$ of functions $\underline{u}(r)$, $\overline{u}(r); 0 \leq r \leq 1$ which satisfying the following requirements:

i. $\underline{u}(r)$ is a bounded monotonic increasing left continuous function,

ii. $\overline{u}(r)$ is a bounded monotonic decreasing left continuous function,

iii. $\underline{u}(r) \leq \tilde{u}(r); 0 \leq r \leq 1$. For arbitrary $u = (u, \tilde{u})$, $v = (\bar{v}, \tilde{v})$ and $k > 0$, we define addition $(u + v)$ and multiplication by $k$ as:

$$u + v(r) = u(r) + v(r), \quad (2.1)$$

$$\underline{u} + \overline{v}(r) = \underline{u}(r) + \overline{v}(r), \quad (2.2)$$

$$\underline{k u}(r) = k \underline{u}(r), \overline{k u}(r) = k \overline{u}(r) \text{ if } k \geq 0, \quad (2.3)$$

$$\underline{k u}(r) = k \underline{u}(r), \overline{k u}(r) = k \overline{u}(r) \text{ if } k < 0. \quad (2.4)$$

2.3. Definition

For arbitrary fuzzy numbers $u = (u, \tilde{u})$ and $v = (\bar{v}, \tilde{v})$ the quantity

$$d(u,v) = \sup_{0 \leq r \leq 1} \left[ \max \left\{ |u(r) - v(r)|, |\tilde{u}(r) - \tilde{v}(r)| \right\} \right] \quad (2.5)$$
is the distance between $u$ and $v$ which also known as the Hausdorff distance between fuzzy numbers given by $d : \mathbb{E} \times \mathbb{E} \rightarrow \mathbb{R}_+ \cup \{0\}$. Then, it can be see that $d$ is a metric in $\mathbb{E}$ and has the following properties:

i. $d(u + w, v + w) = d(u, v) \quad \forall u, v, w \in \mathbb{E},$

ii. $d(\lambda u, \lambda v) = |\lambda| d(u, v) \quad \forall \lambda \in \mathbb{R}, u, v \in \mathbb{E},$

iii. $d(u + v, w + e) \leq d(u, w) + d(v, e) \quad \forall u, v, w \in \mathbb{E},$

iv. $(d, \mathbb{E})$ is a complete metric space.

The parametric form of FFIE-2 can be represented as follows [2]

$$x(s, r) = f(s, r) + \lambda \int_a^b U(t, r) dt$$

(2.6)

$$\bar{x}(s, r) = \bar{f}(s, r) + \lambda \int_a^b \bar{U}(t, r) dt$$

(2.7)

where

$$\bar{U}(t, r) = \begin{cases} K(s, t), x(t, r), K(s, t) \geq 0 \\ K(s, t), x(t, r), K(s, t) < 0 \end{cases}$$

and

$$\bar{U}(t, r) = \begin{cases} K(s, t), \bar{x}(t, r), K(s, t) \geq 0 \\ K(s, t), \bar{x}(t, r), K(s, t) < 0 \end{cases}$$

for each $0 \leq r \leq 1$ and $t \in [a, b]$.

3. Simpson’s 1/3 Approximation Equation

In this section, we use discretization scheme based on Simpson’s 1/3 quadrature rule to derive the approximation equation for linear FFIE-2. The general quadrature scheme based on NC equations over the fuzzy functions, $y(t)$ and $\bar{y}(t)$ is given as follows [15]

$$\left(\int_a^b y(t, r) dt\right) = \sum_{j=0}^n A_j \bar{y}(t_j, r) + e_n(y)$$

(3.1)

$$\left(\int_a^b \bar{y}(t, r) dt\right) = \sum_{j=0}^n A_j \bar{y}(t_j, r) + e_n(y)$$

(3.2)

where $A_j$ is an independent numerical coefficient to the functions $y(t)$ and $\bar{y}(t)$, $t_j$ is the abcissa of the partition points of integrations on the interval $[a, b]$ and $e_n(y)$ is the truncation error of equation (3.1) and (3.2). To simplify us in order to derive a linear system from the discretization process of the integral equations, both equation (3.1) and (3.2) needs to be represented in a general form as follows

$$\int_a^b y(t, r) dt = \sum_{j=0}^n A_j y(t_j, r) + e_n(y)$$

(3.3)

The general definition of Simpson’s 1/3 integral equation of $y(t, r)$ in equation (3.3) on interval $[t_0, t_{n+2}]$ is stated as follows

$$\int_a^{t_{n+2}} y(t, r) dt = \frac{h}{3} \left(y_{i, r} + 4y_{i+1, r} + y_{i+2, r}\right)$$

(3.4)

where the constant step, $h$ is defined as follows
\[ h = \frac{b - a}{n} \]  

(3.5)

and \( n \) is number of subintervals in the interval \([a, b]\) which can be divided into subinterval \([x_0, x_1, x_2, \ldots, x_n]\) as \( a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b \). Finite grid networks and uniform distribution of node points can be represented in figure 1 [15]

**Figure 1.** Finite grid networks and the uniform distribution of node points on interval \([a, b]\).

Referring to the coefficient of equation (3.4), \( A_j \) in equation (3.3) satisfies the following relations [16]

\[
A_j = \begin{cases} 
\frac{1}{3} h, & j = 0, n \\
\frac{4}{3} h, & j = 1, 3, 5, \ldots, n - 1 \\
\frac{2}{3} h, & \text{otherwise}
\end{cases}
\]

(3.6)

Let \( F(s, t) = k(s, t) \overline{x}(t, r) \), \( \overline{x} = (\overline{x}(s, r), \overline{x}(s, r)) \) and \( \lambda = 1 \), both equation (2.6) and (2.7) can be written as the following equations

\[
x(s, r) = f(s, r) + \int_0^t F(s, t) dt \\
\overline{x}(s, r) = \overline{f}(s, r) + \int_0^t F(s, t) dt
\]

(3.7)

(3.8)

For the simplification, equation (3.7) and (3.8) is combined into a general form as follows

\[
x(s, r) = f(s, r) + \int_0^t F(s, t) dt
\]

(3.9)

where \( n = 4 \). Now, we apply equation (3.4) into (3.9) to get the following equation

\[
x(s, r) = f(s, r) + \frac{h}{3} \left[ k(s, t_0) x(t_0, r) + \cdots + 4k(s, t_4) x(t_4, r) \right]
\]

(3.10)

Next, by simplifying equation (3.7), we rewrite the equation as follows

\[
x(s) = f(s) + \frac{h}{3} \left[ k(s, t_0) x(t_0) + \cdots + 4k(s, t_4) x(t_4) \right]
\]

(3.11)

By considering five node points, \( s = s_0, s = s_1, s = s_2, s = s_3 \) and \( s = s_4 \), each node point will have the following approximation equation respectively

\[
(1 - A_0(k_{0,0}))x_0 - A_0(k_{0,1})x_1 - A_2(k_{0,2})x_2 - A_3(k_{0,3})x_3 - A_4(k_{0,4})x_4 = f_0
\]

(3.12)

\[
- A_0(k_{1,0})x_0 + (1 - A_1(k_{1,1}))x_1 - A_2(k_{1,2})x_2 - A_3(k_{1,3})x_3 - A_4(k_{1,4})x_4 = f_1
\]

(3.13)

\[
- A_0(k_{2,0})x_0 - A_1(k_{2,1})x_1 + (1 - A_2(k_{2,2}))x_2 - A_3(k_{2,3})x_3 - A_4(k_{2,4})x_4 = f_2
\]

(3.14)

\[
- A_0(k_{3,0})x_0 - A_1(k_{3,1})x_1 - A_2(k_{3,2})x_2 + (1 - A_3(k_{3,3}))x_3 - A_4(k_{3,4})x_4 = f_3
\]

(3.15)

\[
- A_0(k_{4,0})x_0 - A_1(k_{4,1})x_1 - A_2(k_{4,2})x_2 - A_3(k_{4,3})x_3 + (1 - A_4(k_{4,4}))x_4 = f_4
\]

(3.16)
Based on the linear system in equation (3.12) to (3.16), this linear system for \((n + 1)\) node points in figure 1 can be represented in a general matrix form as follows

\[ A\tilde{x} = \tilde{f} \]  

(3.17)

where

\[
A = \begin{bmatrix}
1 - A_0(k_{0,0}) & -A_1(k_{0,1}) & -A_2(k_{0,2}) & \cdots & -A_n(k_{0,n}) \\
-A_0(k_{1,0}) & 1 - A_1(k_{1,1}) & -A_2(k_{1,2}) & \cdots & -A_n(k_{1,n}) \\
-A_0(k_{2,0}) & -A_1(k_{2,1}) & 1 - A_2(k_{2,2}) & \cdots & -A_n(k_{2,n}) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-A_0(k_{n,0}) & -A_1(k_{n,1}) & -A_2(k_{n,2}) & \cdots & 1 - A_n(k_{n,n}) 
\end{bmatrix},
\]

\[ \tilde{x} = [x_0, x_1, x_2, \cdots, x_n]^T, \]

\[ \tilde{f} = [f_0, f_1, f_2, \cdots, f_n]^T. \]

4. Formulation of the Successive Over-Relaxation Method

Next, SOR iterative method is applied to solve the generated system of linear equations (3.17) over some numerical examples in Section 5. The coefficient matrix \(A\) of linear system (3.17) can be decomposed into \(L\) (strictly lower triangular matrix), \(D\) (diagonal matrix) and \(U\) (strictly upper triangular matrix) as follows

\[ A = L + D + U \]

The general SOR iterative method can be defined as follows [15,17]

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

(4.1)

where \(\omega\) is a weighted parameter. Then, the value of \(\omega\) within \(\pm 0.01\) is obtained by running the program for different values of \(\omega\). Next, the value of \(\omega\) with the minimum number of iterations will be chosen and recorded in the tables. The general algorithm of SOR iterative method based on formulation (4.1) to solve linear FFIE-2 which is described as follows [15,17,18,19]

4.1. Algorithm of SOR iterative method

i. Set the initial value \(\tilde{x}^{(0)} = 0, \varepsilon = 10^{-10}\)

ii. Calculate matrix \(A\) and vector \(\tilde{f}\).

iii. For \(i = 1,2,3,\ldots,n\), calculate

\[
(1 - \omega)\tilde{x}_i^{(k)} + \omega \sum_{j=0}^{n} A_{i,j}\tilde{x}_j^{(k)} + \omega f_i \big/ A_{i,i}, \quad i = 0
\]

\[
\tilde{x}_i^{(k+1)} \leftarrow (1 - \omega)\tilde{x}_i^{(k)} + \omega \sum_{j=0}^{n} A_{i,j}\tilde{x}_j^{(k+1)} + \omega f_i \big/ A_{i,i}, \quad i = n
\]

\[
(1 - \omega)\tilde{x}_i^{(k)} + \omega \sum_{j=0}^{n} A_{i,j}\tilde{x}_j^{(k+1)} + \omega \sum_{j=0}^{n} A_{i,j}\tilde{x}_j^{(k)} + \omega f_i \big/ A_{i,i}, \quad \text{others}
\]

iv. Convergence test \(\tilde{x}_i^{(k+1)} - \tilde{x}_i^{(k)} \leq \varepsilon = 10^{-10}\). If the convergence criterion is satisfied, go to step v and otherwise go back to step iii.

v. Display approximate solutions.
5. Numerical Examples

In this section, we present some numerical examples of FFIE-2 problems to compare the performances of the Jacobi, GS and SOR iterative methods.

5.1. Example 1
Consider the following FFIE-2 problem [13]
\[
\begin{align*}
\Delta f(s,r) &= - \frac{1}{52} r(5-52s + 2s^2) \\
\tilde{f}(s,r) &= \frac{1}{52} (r-2)(5-52s + 2s^2)
\end{align*}
\]
and kernel
\[
k(s,t) = \frac{5^2 + r^2 + 2}{13}, \quad 0 \leq s, t \leq 1 \text{ and } \lambda = 1
\]
and \(a = 0, b = 1\). The exact solutions of the problem are
\[
\begin{align*}
\hat{x}(s,r) &= rs \\
\tilde{x}(s,r) &= (2-r)s
\end{align*}
\]

5.2. Example 2
Consider the following FFIE-2 problem [13]
\[
\begin{align*}
\Delta f(s,r) &= \frac{2}{3} (2+r)s \\
\tilde{f}(s,r) &= -\frac{2}{3} (r-4)s
\end{align*}
\]
and kernel
\[
k(s,t) = st, \quad 0 \leq s, t \leq 1 \text{ and } \lambda = 1
\]
and \(a = 0, b = 1\). The exact solutions of the problem are
\[
\begin{align*}
\hat{x}(s,r) &= (2+r)s \\
\tilde{x}(s,r) &= (4-r)s
\end{align*}
\]

5.3. Example 3
Consider the following FFIE-2 problem [20]
\[
\begin{align*}
\Delta f(s,r) &= rs(r^4 + 2)\left(3\cos(1-s) + 5\sin(1-s) - 6\cos(s) + s^2\right) \\
\tilde{f}(s,r) &= -3s(r^3 - 2)\left(3\cos(1-s) + 5\sin(1-s) - 6\cos(s) + s^2\right)
\end{align*}
\]
and kernel
\[
k(s,t) = s \cos(t-s), \quad 0 \leq s, t \leq 1 \text{ and } \lambda = 1
\]
and \(a = 0, b = 1\). The exact solutions of the problem are
\[
\begin{align*}
\hat{x}(s,r) &= s^3(r^5 + 2r) \\
\tilde{x}(s,r) &= s^3(6 - 3r^3)
\end{align*}
\]

By using Jacobi iterative method as a control method, we compare the results between GS and SOR iterative methods with the Jacobi. Three parameters which are the number of iterations (K), execution time (Time) in second and Hausdorff distance (HD) are used to compare the performances of the proposed method. The value of convergence test is fixed at tolerance error of \(\varepsilon = 10^{-10}\). Next, the simulations are carried out on several mesh sizes (M) which are 256, 512, 1024, 2048 and 4096.
Moreover, the simulations are run on the value of $r = 1.0$, 0.6 and 0.3 for $0 \leq r \leq 1$ in each example. The numerical results of the Jacobi, GS and SOR iterative methods for example 1, 2 and 3 are recorded in table 1, 2 and 3 respectively. Then, the percentages of iterations number and execution time for the GS and SOR compared with Jacobi iterative method is recorded in table 4.

6. Conclusions

In this paper, we present SOR iterative method for solving linear FFIE-2 by using Simpson’s 1/3 rule. In terms of number of iterations, the numerical result for example 1 show that GS and SOR declined approximately 29.03-32.26% and 48.39-50.00% respectively compared to Jacobi. For example 2, the result shows that GS and SOR decreased 39.13% and 56.52% respectively compared to Jacobi. Meanwhile, the result for example 3 shows that GS and SOR decreased approximately 40.63-42.19% and 60.32-62.50% respectively compared to Jacobi. In terms of execution time, the numerical result for example 1 show that GS and SOR faster about 26.92-50.00% and 42.31-62.50% respectively compared to Jacobi. For example 2, the result shows that GS and SOR faster about 33.33-49.33% and 53.12-64.00% respectively compared to Jacobi. Meanwhile, the result for example 3 show that GS and SOR faster about 36.97-50.72% and 55.73-62.45% respectively compared to Jacobi. In terms of Hausdorff distance, the overall result recorded that Hausdorff distance for all examples almost zero. Therefore, we conclude that the numerical results obtained are accurate and can be accepted. Overall, it can be clearly concluded that the number of iterations declined by using GS and SOR methods. In terms of execution time, it can be concluded that GS and SOR methods are much faster than Jacobi method. Finally, we conclude that SOR method is a better method compared to the Jacobi and GS method in terms of iterations number and execution time.

Table 1. Comparison of the number of iterations, execution time and Hausdorff distance of the iterative methods at $r = 1.0$, 0.6 and 0.3 for example 1.

| M    | Methods | $r = 1.0$     |          | $r = 0.6$     |          | $r = 0.3$     |          |
|------|---------|---------------|----------|---------------|----------|---------------|----------|
|      |         | K             | Time     | HD            |          | K             | Time     | HD            |          |
| 256  | Jacobi  | 32            | 0.08     | 7.40E-12      |          | 31            | 0.07     | 2.15E-11      |          |
|      | GS      | 22            | 0.04     | 1.15E-12      |          | 22            | 0.05     | 1.61E-12      |          |
|      | SOR     | 16            | 0.03     | 2.93E-12      |          | 16            | 0.04     | 4.10E-12      |          |
| 512  | Jacobi  | 32            | 0.26     | 7.61E-12      |          | 31            | 0.30     | 2.20E-11      |          |
|      | GS      | 22            | 0.19     | 1.19E-12      |          | 22            | 0.19     | 1.67E-12      |          |
|      | SOR     | 16            | 0.15     | 2.95E-12      |          | 16            | 0.14     | 4.13E-12      |          |
| 1024 | Jacobi  | 32            | 1.01     | 7.71E-12      |          | 31            | 1.02     | 2.23E-11      |          |
|      | GS      | 22            | 0.70     | 1.22E-12      |          | 22            | 0.72     | 1.71E-12      |          |
|      | SOR     | 16            | 0.52     | 2.97E-12      |          | 16            | 0.53     | 4.16E-12      |          |
| 2048 | Jacobi  | 32            | 4.01     | 7.77E-12      |          | 31            | 3.91     | 2.25E-11      |          |
|      | GS      | 22            | 2.75     | 1.23E-12      |          | 22            | 2.78     | 1.72E-12      |          |
|      | SOR     | 16            | 2.02     | 2.98E-12      |          | 16            | 2.04     | 4.17E-12      |          |
| 4096 | Jacobi  | 32            | 16.20    | 7.79E-12      |          | 31            | 15.32    | 2.25E-11      |          |
|      | GS      | 22            | 10.92    | 1.24E-12      |          | 22            | 10.61    | 1.73E-12      |          |
|      | SOR     | 16            | 8.01     | 2.98E-12      |          | 16            | 7.98     | 4.18E-12      |          |
### Table 2. Comparison of the number of iterations, execution time and Hausdorff distance of the iterative methods at $r = 1.0, 0.6$ and $0.3$ for example 2.

| M   | Methods | $r = 1.0$ | $r = 0.6$ | $r = 0.3$ |
|-----|---------|-----------|-----------|-----------|
|     |         | K  | Time | HD    | K  | Time | HD    | K  | Time | HD    |
| 256 | Jacobi  | 46 | 0.09 | 2.83E-11 | 46 | 0.09 | 3.21E-11 | 46 | 0.09 | 3.49E-11 |
|     | GS      | 28 | 0.06 | 9.96E-11 | 28 | 0.05 | 1.13E-11 | 28 | 0.06 | 1.23E-11 |
|     | SOR     | 20 | 0.04 | 3.00E-12 | 20 | 0.04 | 3.39E-12 | 20 | 0.04 | 3.69E-12 |
| 512 | Jacobi  | 46 | 0.32 | 3.01E-11 | 46 | 0.32 | 3.41E-11 | 46 | 0.32 | 3.71E-11 |
|     | GS      | 28 | 0.20 | 1.07E-11 | 28 | 0.20 | 1.22E-11 | 28 | 0.21 | 1.32E-11 |
|     | SOR     | 20 | 0.15 | 2.89E-12 | 20 | 0.14 | 3.27E-12 | 20 | 0.14 | 3.56E-12 |
| 1024| Jacobi  | 46 | 1.24 | 3.09E-11 | 46 | 1.50 | 3.51E-11 | 46 | 1.23 | 3.82E-11 |
|     | GS      | 28 | 0.76 | 1.11E-11 | 28 | 0.76 | 1.26E-11 | 28 | 0.76 | 1.37E-11 |
|     | SOR     | 20 | 0.55 | 2.85E-12 | 20 | 0.54 | 3.23E-12 | 20 | 0.55 | 3.51E-12 |
| 2048| Jacobi  | 46 | 4.90 | 3.14E-11 | 46 | 4.90 | 3.56E-11 | 46 | 4.89 | 3.87E-11 |
|     | GS      | 28 | 2.99 | 1.13E-11 | 28 | 2.98 | 1.29E-11 | 28 | 2.99 | 1.40E-11 |
|     | SOR     | 20 | 2.14 | 2.84E-12 | 20 | 2.16 | 3.21E-12 | 20 | 2.16 | 3.50E-12 |
| 4096| Jacobi  | 46 | 19.25 | 3.16E-11 | 46 | 18.89 | 3.59E-11 | 46 | 19.31 | 3.90E-11 |
|     | GS      | 28 | 11.75 | 1.15E-11 | 28 | 11.83 | 1.30E-11 | 28 | 11.71 | 1.41E-11 |
|     | SOR     | 20 | 8.46 | 2.83E-12 | 20 | 8.53 | 3.21E-12 | 20 | 8.47 | 3.49E-12 |

### Table 3. Comparison of the number of iterations, execution time and Hausdorff distance of the iterative methods at $r = 1.0, 0.6$ and $0.3$ for example 3.

| M   | Methods | $r = 1.0$ | $r = 0.6$ | $r = 0.3$ |
|-----|---------|-----------|-----------|-----------|
|     |         | K  | Time | HD    | K  | Time | HD    | K  | Time | HD    |
| 256 | Jacobi  | 64 | 1.10 | 1.20E-10 | 64 | 1.13 | 1.65E-10 | 63 | 1.07 | 1.82E-10 |
|     | GS      | 38 | 0.65 | 8.12E-11 | 37 | 0.64 | 1.45E-10 | 37 | 0.65 | 1.60E-10 |
|     | SOR     | 24 | 0.43 | 7.57E-11 | 25 | 0.44 | 1.24E-10 | 25 | 0.44 | 1.37E-10 |
| 512 | Jacobi  | 64 | 4.29 | 6.07E-11 | 64 | 4.37 | 5.51E-11 | 63 | 4.29 | 6.09E-11 |
|     | GS      | 38 | 2.57 | 1.91E-11 | 37 | 2.55 | 3.41E-11 | 37 | 2.55 | 3.77E-11 |
|     | SOR     | 24 | 1.67 | 1.34E-11 | 25 | 1.73 | 8.98E-12 | 25 | 1.73 | 1.03E-11 |
| 1024| Jacobi  | 64 | 16.46 | 5.83E-11 | 64 | 17.45 | 5.24E-11 | 63 | 17.50 | 5.47E-11 |
|     | GS      | 38 | 10.19 | 1.59E-11 | 37 | 9.89 | 2.84E-11 | 37 | 8.82 | 3.14E-11 |
|     | SOR     | 24 | 6.42 | 9.48E-12 | 25 | 6.86 | 4.88E-12 | 5  | 6.69 | 2.80E-12 |
| 2048| Jacobi  | 64 | 65.76 | 5.88E-11 | 64 | 67.45 | 5.29E-11 | 63 | 65.06 | 5.49E-11 |
|     | GS      | 38 | 41.45 | 1.60E-11 | 37 | 38.51 | 2.85E-11 | 37 | 36.29 | 3.15E-11 |
|     | SOR     | 24 | 25.74 | 9.25E-12 | 25 | 27.05 | 4.78E-12 | 25 | 26.19 | 3.02E-12 |
| 4096| Jacobi  | 64 | 269.23 | 5.92E-11 | 64 | 257.16 | 5.32E-11 | 63 | 242.15 | 5.53E-11 |
|     | GS      | 38 | 153.95 | 1.61E-11 | 37 | 126.73 | 2.87E-11 | 37 | 126.92 | 3.17E-11 |
|     | SOR     | 24 | 101.09 | 9.25E-12 | 25 | 107.87 | 4.77E-12 | 25 | 107.21 | 3.05E-12 |

8
Table 4. Percentages of the number of iterations and execution time for the GS and SOR iterative methods compared with Jacobi iterative method.

| Methods | Number of iterations | Example 1 | Example 2 | Example 3 |
|---------|----------------------|-----------|-----------|-----------|
| GS      | 29.03-32.26%         | 39.13%    | 40.63-42.19% |
| SOR     | 48.39-50.00%         | 56.52%    | 60.32-62.50% |

| Methods | Execution time | Example 1 | Example 2 | Example 3 |
|---------|---------------|-----------|-----------|-----------|
| GS      | 26.92-50.00%  | 33.33-49.33% | 36.97-50.72% |
| SOR     | 42.31-62.50%  | 53.12-64.00% | 55.73-62.45% |

References

[1] Shamivand M M, Shahsavaran A and Tari S M 2011 Int. J. of Contemporary Mathematics Sciences 6 pp 535-543
[2] Barkhordary M, Kiani N A and Bozorgmanesh A R 2008 Int. J. of Open Problems in Computer Science and Mathematics 1 pp 149-159
[3] Jahantigh M, Allahviranloo T and Otandi M 2008 Applied Mathematical Sciences 2 pp 33-46
[4] Salehi P and Nejatiyan M 2013 Int. J. of Industrial Mathematics 5 pp 161-165
[5] Mirzaee F, Paripour M and Komak Y M 2012 J. of Hyperstructure 1 pp 46-60
[6] Mirzaee F 2014 Ain Shams Engineering J. 5 pp 631-636
[7] Molabahrami A, Shidfar S and Ghyasi A 2011 Computer and Mathematics with Applications 61 pp 2754-2761
[8] Abbasbandy S, Babolian E and Alavi M 2007 Chaos, Solitons and Fractals 31 pp 138-146
[9] Amawi M and Qatanani N 2016 Research J. of Applied Sciences 11 pp 1559-1568
[10] Fadravi H H, Buzhabadi R and Nik H S 2014 Alexandria Engineering J. 53 pp 249-257
[11] Ezzati R and Ziari S 2013 Applied Mathematics and Computation 225 pp 33-42
[12] Otadi M and Mosleh M 2014 Iranian J. of Fuzzy System 11 pp 135-145
[13] Gholam A M and Ezzati R 2017 TWMS J. of Pure Applied Mathematics 8 pp 121-147
[14] Salahshour S, Khezerloo M, Hajighasemi S and Khowasany M 2012 Int. J. of Industrial Mathematics 4 pp 21-29
[15] Muthuvalu M S and Sulaiman J 2011 Discovering Mathematics 33 pp 49-57
[16] Muthuvalu M S and Sulaiman J 2011 Int. J. Open Problem Computational Mathematics 4 pp 201-213
[17] Youssef I 2012 J. of Mathematics and Statistics 8 pp 176-184
[18] Muthuvalu M S and Sulaiman J 2011 Australian J. of Basic and Applied Sciences 5 pp 18-24
[19] Dahalan A A, Aziz N S A and Sulaiman J 2016 J. of Engineering and Applied Sciences 11 pp 1456-1463
[20] Mirzaee F, Paripour M and Komak Y M 2014 J. Science Engineering 39, pp 3969-3978