A Preconditioning Algorithm for the Positive Solution of Fully Fuzzy Linear System

K. Dookhitram\textsuperscript{1,*}, M. S. Sunhaloo\textsuperscript{1}, N. Rambeerich\textsuperscript{1}, A. A. I. Peer\textsuperscript{1}, A. A. E. F. Saib\textsuperscript{1}

(1) Department of Applied Mathematical Sciences, School of Innovative Technologies and Engineering, University of Technology, Mauritius, La Tour Koenig, Pointe-aux-Sables, Republic of Mauritius

Abstract
A linear system of equations is called a fully fuzzy linear system (FFLS) if all the quantities of this system are fuzzy numbers. We consider the positive solution of FFLS, where the modal value (center) matrix is positive definite and we develop a new approximate procedure based on preconditioning. We observe from the numerical results that our method is more accurate than the iterative Jacobi, Gauss-Seidel and Successive Over-Relaxation (SOR) methods when finding approximate solutions of FFLS.

Keywords: Fully fuzzy linear system; Trapezoidal fuzzy number; Positive definite matrices; Conjugate gradient algorithm; Preconditioning techniques.

1 Introduction

When information is imprecise and only some vague knowledge about the actual values of the parameters is available, it is convenient to make use of fuzzy numbers [13]. One of the major applications of fuzzy number arithmetic is solving of linear systems whose parameters are all or partially represented by fuzzy numbers. System of fully fuzzy linear equations has been of growing interest in recent years. Many financial problems, for instance option valuation or portfolio selection, are modeled as linear systems. In the presence of uncertainty, those financial models are said to operate as FFLS [9]. In this paper, the term fuzzy matrix is of the most importance concept, and we follow the definition proposed in [4], that is, a matrix with fuzzy numbers as its elements.

\*Corresponding author. Email address: kdookhitram@umail.utm.ac.mu Tel: +2302075250
In [5] a general model is introduced for solving $n \times n$ fuzzy linear system whose coefficient matrix is crisp and the right hand vector to be an arbitrary fuzzy vector. Friedman et al. solved the fuzzy linear system by first reducing it to a $2n \times 2n$ crisp linear system. Review about some methods for solving these systems can be found in [10]. Here we consider another kind of fuzzy linear systems where all the parameters include fuzzy numbers. Recently, in [1, 2] computational methods such as Cramer’s rule, Gaussian elimination method, LU decomposition method and Adomian decomposition method have been used for solving FFLS. Iterative techniques for the solution of FFLS is presented in [3], whereby techniques such as Richardson, Jacobi, Jacobi over relaxation, Gauss-Seidel, SOR, accelerated over relaxation, symmetric and unsymmetrical successive over relaxation and extrapolated modified Aitken are studied. For other methods to solve FFLS, one may refer to [8].

Here, we consider FFLS of the form $\tilde{\Psi} \otimes \tilde{\xi} = \tilde{\phi}$, where $\tilde{\Psi} = (\Phi, \Lambda, \Theta)$, with the model value (center) matrix $\Phi$ to be positive definite. We consider the use of the conjugate gradient method with preconditioning technique for approximating the positive solution of the above defined FFLS. We demonstrate with examples that the preconditioned algorithm converges to the exact solution more rapidly than the iterative Jacobi, Gauss-Seidel and SOR methods. This paper is structured as follows: In the next section, we give some preliminaries concerning fuzzy sets theory. In section 3, the new procedure based on preconditioning is introduced. Numerical examples are presented in section 4 to illustrate efficiency of the method.

## 2 Preliminaries

In this section, we present some brief backgrounds and notions of fuzzy sets theory [4, 10].

### 2.1 Fuzzy Numbers

**Definition 2.1.** Assume $\Omega$ to be a universal set, and then a fuzzy subset $\tilde{\Phi}$ of $\Omega$ is defined by its membership function 

$$
\mu_{\tilde{\Phi}} : \Omega \rightarrow [0, 1],
$$

where the value of $\mu_{\tilde{\Phi}}(\omega)$ at $\omega$ shows the grade of membership of $\omega$ in $\tilde{\Phi}$.

A fuzzy subset $\tilde{\Phi}$ can be characterized as a set of ordered pairs of element $\omega$ and grade $\mu_{\tilde{\Phi}}(\omega)$ and is often described as

$$
\tilde{\Phi} = \{(\omega, \mu_{\tilde{\Phi}}(\omega)) : \omega \in \Omega\}.
$$

A fuzzy set $\tilde{\Phi}$ in $\Omega$ is said to be normal if there exists $\omega_0 \in \Omega$ such that $\mu_{\tilde{\Phi}}(\omega_0) = 1$.

**Definition 2.2.** An arbitrary fuzzy number $\tilde{\eta}(r)$ may be represented by a pair of functions

$$
\tilde{\eta}(r) = (\eta(r), \bar{\eta}(r)), \ r \in [0, 1],
$$

which satisfies the following conditions:

(i) $\eta(r)$ is a bounded continuous non-decreasing function over $[0, 1]$.

(ii) $\bar{\eta}(r)$ is a bounded continuous non-increasing function over $[0, 1]$. 


(iii) $\eta(r) \leq \overline{\eta}(r)$, $r \in [0, 1]$.

The set of all fuzzy numbers $\overline{\eta}(r)$ is a convex cone which is embedded isomorphically and isometrically into a Banach space. When $\overline{\eta}(r) = \overline{\eta}(r)$, the fuzzy number is simply referred to as a crisp number.

The trapezoidal fuzzy number is defined as $\eta(\varsigma, \tau, \sigma, \beta)$, where $\varsigma$ and $\tau$ are known as the defuzzifiers; and $\sigma$ and $\beta$ are the left and right fuzziness (spreads), respectively. The case $\varsigma = \tau$ is referred to as the triangular fuzzy number and is written as $\eta(\varsigma, \sigma, \beta)$. For the trapezoidal fuzzy number, the parametric form is given by

$$\eta(r) = \varsigma - \sigma + \sigma r$$

and

$$\overline{\eta}(r) = \tau + \beta - \beta r.$$

**Definition 2.3.** A fuzzy number $\tilde{\eta}$ is of LR-type if there exists reference functions $L$ (for left), $R$ (for right) and scalars $\sigma > 0$, $\beta > 0$ with $\eta = (\varsigma, \sigma, \beta)$, if its membership function has the following forms:

$$\mu_{\tilde{\eta}}(\omega) = \begin{cases} L\left(\frac{\omega - \varsigma}{\sigma}\right), & \text{if } \omega \leq \varsigma; \\ 1, & \text{if } \omega = \varsigma; \\ R\left(\frac{\omega - \varsigma}{\beta}\right), & \text{if } \omega \geq \varsigma; \end{cases}$$

where

$$L, R : [0, +\infty) \rightarrow [0, 1]$$

with $L(x) = L(-x)$, $R(x) = R(-x)$, $L(0) = 1$ and $R(0) = 1$.

We say that $\tilde{\eta} = (\varsigma, \sigma, \beta)$ is positive (or negative), denoted by $\tilde{\eta} > 0$ (or $\tilde{\eta} < 0$), if its membership function $\mu_{\tilde{\eta}}(\omega) = 0$, for all $\omega < 0$ (or $\omega > 0$). Further, two fuzzy numbers $\tilde{\eta} = (\varsigma, \sigma, \beta)$ and $\tilde{\zeta} = (\varrho, \gamma, \kappa)$ are equal if $\varsigma = \varrho$, $\sigma = \gamma$ and $\beta = \kappa$.

On the other hand, the fuzzy matrix is defined as:

**Definition 2.4.** A matrix $\tilde{\Psi} = (\tilde{\psi}_{ij})$ is called a fuzzy matrix if each element in $\tilde{\Psi}$ is a fuzzy number.

The matrix $\tilde{\Psi}$ is positive if each of its elements is positive. Also, the $n \times n$ matrix $\tilde{\Psi}$ may be represented as

$$\tilde{\Psi} = (\Phi, \Lambda, \Theta),$$

where the crisp matrices $\Phi = (\phi_{ij})$, $\Lambda = (\lambda_{ij})$ and $\Theta = (\theta_{ij})$ are called the model value (center) matrix and the left and right spread matrices, respectively.

### 2.2 Arithmetic Operations on Fuzzy Numbers

Suppose $\mathcal{F}$ be the set of LR-fuzzy numbers and let $\tilde{\eta} = (\varsigma, \sigma, \beta)$ and $\tilde{\zeta} = (\varrho, \gamma, \kappa)$ be two fuzzy numbers in $\mathcal{F}$. The following operations are valid [9]:

(a) $\tilde{\eta} \oplus \tilde{\zeta} = (\varsigma + \varrho, \sigma + \gamma, \beta + \kappa)$.
(b) \(-\tilde{\eta} = -(\zeta, \sigma, \beta) = (-\zeta, \beta, \sigma)\).

(c) (i) If \(\tilde{\eta} > 0\) and \(\tilde{\zeta} > 0\), then
\[
\tilde{\eta} \otimes \tilde{\zeta} = (\zeta \varrho, \zeta \gamma + \varrho \sigma, \zeta \kappa + \varrho \beta).
\]

(ii) If \(\tilde{\eta} < 0\) and \(\tilde{\zeta} > 0\), then
\[
\tilde{\eta} \otimes \tilde{\zeta} = (\zeta \varrho, -\zeta \gamma + \varrho \sigma, -\zeta \kappa + \varrho \beta).
\]

(iii) If \(\tilde{\eta} < 0\) and \(\tilde{\zeta} < 0\), then
\[
\tilde{\eta} \otimes \tilde{\zeta} = (\zeta \varrho, -\zeta \gamma - \varrho \sigma, -\zeta \kappa - \varrho \beta).
\]

(d) If \(\nu\) is any scalar, then \(\nu \otimes \tilde{\eta}\) is defined as
\[
\nu \otimes \tilde{\eta} = \begin{cases} 
(\nu \zeta, \nu \sigma, \nu \beta), & \nu \geq 0; \\
(\nu \zeta, -\nu \beta, -\nu \sigma), & \nu < 0.
\end{cases}
\]

3 Fully Fuzzy Linear Systems

In this section we review the solution procedure for FFLS. Consider the \(n \times n\) FFLS of the form
\[
(\tilde{\psi}_{11} \otimes \tilde{\xi}_1) \oplus (\tilde{\psi}_{12} \otimes \tilde{\xi}_2) \oplus \cdots \oplus (\tilde{\psi}_{1n} \otimes \tilde{\xi}_n) = \tilde{\varphi}_1,
\]
\[
(\tilde{\psi}_{21} \otimes \tilde{\xi}_1) \oplus (\tilde{\psi}_{22} \otimes \tilde{\xi}_2) \oplus \cdots \oplus (\tilde{\psi}_{2n} \otimes \tilde{\xi}_n) = \tilde{\varphi}_2,
\]
\[
\vdots
\]
\[
(\tilde{\psi}_{n1} \otimes \tilde{\xi}_1) \oplus (\tilde{\psi}_{n2} \otimes \tilde{\xi}_2) \oplus \cdots \oplus (\tilde{\psi}_{nn} \otimes \tilde{\xi}_n) = \tilde{\varphi}_n,
\]
which may be represented as
\[
\tilde{\Psi} \otimes \tilde{\xi} = \tilde{\varphi},
\]
in matrix form, with \(\tilde{\Psi} = (\tilde{\psi}_{ij})\), \(1 \leq i, j \leq n\) and \(\tilde{\xi}, \tilde{\varphi} \in \mathcal{F}\). Let \(\tilde{\Psi} = (\Phi, \Lambda, \Theta), \tilde{\xi} = (\xi_1, \xi_2, \xi_3)\) and \(\tilde{\varphi} = (\varphi_1, \varphi_2, \varphi_3)\), then (3.1) becomes
\[
(\Phi, \Lambda, \Theta) \otimes (\xi_1, \xi_2, \xi_3) = (\varphi_1, \varphi_2, \varphi_3).
\]

Assuming that the coefficient matrix and solution vector are positive in (3.1), then following §2.2 we obtain the crisp systems
\[
\Phi \xi_1 = \varphi_1,
\]
\[
\Phi \xi_2 + \Lambda \xi_1 = \varphi_2,
\]
\[
\Phi \xi_3 + \Theta \xi_1 = \varphi_3.
\]

From (3.2), we find that once a solution to \(\xi_1\), say \(\xi_1^{sol}\), is obtained, solution to \(\xi_2\) and \(\xi_3\) may be computed by solving the crisp linear systems
\[
\Phi \xi_2 = \varphi_2 - \Lambda \xi_1^{sol},
\]
\[
\Phi \xi_3 = \varphi_3 - \Theta \xi_1^{sol}.
\]
In this paper, we assume that the crisp matrix $\Phi$ is positive definite, that is, for all nonzero vectors $\varpi \in \mathbb{C}$, we have $\varpi^T \Phi \varpi > 0$, where $\varpi^T$ denotes the transpose. Positive definite matrices are of great importance in various applications. They are used, for instance, in optimization algorithms and in the construction of a wide variety of linear regression models [6]. We shall mention that the spectrum of a positive definite matrix is the real line. In this respect, next section describes the procedure for solving the crisp linear systems (3.2) for positive definite matrix $\Phi$.

3.1 A New Approach For Solving FFLS

Consider the crisp linear system of the form

$$\Phi \xi = \varphi,$$  \hspace{1cm} (3.3)

where $\Phi$ is a positive definite matrix of order $n$ and $\xi, \varphi \in \mathbb{C}^{n \times 1}$. One of the methods for solving (3.3) is the conjugate gradient method [7] which can be summarized by the following algorithm:

The Conjugate Gradient Algorithm (CGA)

Input: $\Phi, \varphi, \xi^{(0)}, k_{\text{max}}$ and $\epsilon$

Output: An approximation to $\xi$

1. $\gamma^{(0)} = \varphi - \Phi \xi^{(0)}, \rho^{(0)} = \gamma^{(0)}$

2. For $k = 0, 1, 2, \ldots, k_{\text{max}}$

3. $\alpha^{(k)} = \frac{(\rho^{(k)})^T \gamma^{(k)}}{(\rho^{(k)})^T \Phi \rho^{(k)}}$

4. $\xi^{(k+1)} = \xi^{(k)} + \alpha^{(k)} \rho^{(k)}$

5. $\gamma^{(k+1)} = \varphi - \Phi \xi^{(k+1)}$

6. Terminate the iteration process if $\|\gamma^{(k+1)}\|/\|\varphi\| < \epsilon$

7. $\beta^{(k)} = \frac{(\gamma^{(k+1)})^T \gamma^{(k+1)}}{(\gamma^{(k)})^T \gamma^{(k)}}$

8. $\rho^{(k+1)} = \gamma^{(k+1)} + \beta^{(k)} \rho^{(k)}$

9. End-For

In CGA, $\xi^{(0)}$ is the starting vector of the process with $k_{\text{max}}$ and $\epsilon$ being the maximum number of iteration and convergence tolerance, respectively. The quantity $\rho^{(k+1)}$ is known as the conjugate direction and $\gamma^{(k+1)}$ is the residual vector of the linear system (3.3). We note that $(\rho^{(k)})^T$ implies transpose of the vector $(\rho^{(k)})$. It can be verified that the number of iteration required for the residual vector norm to satisfy the convergence tolerance $\epsilon$ is proportion to the square root of the condition number of matrix $\Phi$, that is, $\sqrt{\text{cond}(\Phi)}$.

For the descent method [7], it is well known that the speed of convergence increases when $\text{cond}(\Phi) \rightarrow 1$. In this respect, to accelerate the convergence of CGA, we replace the solution of the system (3.3) by that of an equivalent system

$$\Upsilon^{-1} \Phi \xi = \Upsilon^{-1} \varphi$$

such that $\text{cond}(\Upsilon^{-1} \Phi)$ is as close as possible to one. This process is commonly known as preconditioning.

Using the fact that $\Upsilon^{-1/2}$ is also positive definite when $\Upsilon^{-1}$ is positive definite, the system (3.3) transforms to

$$\hat{\Phi} \chi = \hat{\varphi}$$ and $$\Upsilon^{1/2} \xi = \chi,$$
where $\hat{\Phi} = \Upsilon^{-1/2} \Phi \Upsilon^{-1/2}$ and $\hat{\varphi} = \Upsilon^{-1/2} \varphi$. Letting $\nu = \Upsilon^{-1} \gamma^{(k)}$, then we have the following preconditioned algorithm for the solution of (3.3):

**Preconditioned Conjugate Gradient Algorithm (PCGA)**

Input: $\Phi$, $\varphi$, $\xi^{(0)}$, $k_{\text{max}}$ and $\epsilon$

Output: An approximation to $\xi^{1}$.

1. $\gamma^{(0)} = \varphi - \hat{\Phi} \xi^{(0)}$, $\rho^{(0)} = \gamma^{(0)}$
2. Solve $\Upsilon \rho^{(0)} = \gamma^{(0)}$ and set $\nu^{(0)} = \rho^{(0)}$
3. For $k = 0, 1, 2, \ldots, k_{\text{max}}$
4. \[ \alpha^{(k)} = (\rho^{(k)})^\top \nu^{(k)} \]
5. \[ \xi^{(k+1)} = \xi^{(k)} + \alpha^{(k)} \rho^{(k)} \]
6. $\gamma^{(k+1)} = \varphi - \hat{\Phi} \xi^{(k+1)}$
7. Terminate the iteration process if $\|\gamma^{(k+1)}\|/\|\varphi\| < \epsilon$
8. Solve $\Upsilon \nu^{(k+1)} = \gamma^{(k+1)}$
9. \[ \beta^{(k)} = (\gamma^{(k+1)})^\top \nu^{(k)} \]
10. $\rho^{(k+1)} = \nu^{(k+1)} + \beta^{(k)} \rho^{(k)}$
11. End-For

### 4 Examples

In this section, we consider five examples to demonstrate the efficiency of PCGA for approximating the solution of FFLS. Note that for the first four examples the pseudospectrum of the crisp matrices are provided to demonstrate that the matrix $\Phi$ is positive definite. The EIGTOOL is used for generating the pseudospectrum [12]. All the experiments are run using the MATLAB software and the function `rand()` is used for creating random matrices and vectors. The pair $(k_{\text{max}}, \epsilon)$ is chosen as $(1000, 1E - 10)$. In all the tables, ‘itr’ represents the number of iteration, ‘nc’ implies no convergence in $k_{\text{max}}$ number of iterations, ‘cpu’ denotes the total computational time for computing approximations to $\xi_1$, $\xi_2$ and $\xi_3$; and ‘res’ is the relative residual norm at the last iteration with respect to the crisp linear systems. We note that ‘itr’ $=$ $(x, y, z)$ implies a method takes $x$, $y$, and $z$ number of iteration to find an approximate solution to $\xi_1$, $\xi_2$ and $\xi_3$, respectively. The same hold for the parameter ‘res’.

**Example 4.1.** We consider $\hat{\Psi} = (\Phi, \Lambda, \Theta)$ and $\hat{\varphi} = (\varphi_1, \varphi_2, \varphi_3)$ be a fully fuzzy matrix and a fully fuzzy vector, respectively, with

\[
\Phi = \begin{pmatrix}
15 & 18 & 27 & 37 \\
18 & 30 & 32 & 45 \\
27 & 32 & 78 & 75 \\
37 & 45 & 75 & 102
\end{pmatrix}, \quad \Lambda = \begin{pmatrix}
0.1 & 0.3 & 0.2 & 0.5 \\
0.1 & 0.2 & 0.4 & 0.1 \\
0.2 & 0.3 & 0.5 & 0.1 \\
0.1 & 0.2 & 0.3 & 0.2
\end{pmatrix},
\]

\[
\Theta = \begin{pmatrix}
0.2 & 0.3 & 0.1 & 0.2 \\
0.3 & 0.1 & 0.2 & 0.4 \\
0.3 & 0.1 & 0.2 & 0.1 \\
0.2 & 0.3 & 0.1 & 0.4
\end{pmatrix}, \quad [\varphi_1, \varphi_2, \varphi_3] = \begin{pmatrix}
3 & 0.8 & 0.9 \\
6 & 0.9 & 0.7 \\
2 & 0.4 & 0.8 \\
8 & 0.7 & 0.4
\end{pmatrix}.
\]
Fig. 1 shows the pseudospectrum of the crisp matrices $\Phi$, $\Lambda$ and $\Theta$. We find that the eigenvalues of matrix $\Phi$ are all real and positive. From Table 1 we observe that both CGA and PCGA perform better than the iterative Jacobi, Gauss-Seidel and SOR methods. For comparison purpose, CGA takes five iterations to reach an approximate solution of $\xi_1$, while the Gauss-Seidel and SOR methods take 183 iterations. Also, we find that the Jacobi algorithm does not converge in 1000 iterations. On the other hand, PCGA converges faster than CGA, with a total computational time of $6.00E-3$ as compared to $2.80E-2$.

Table 1

<table>
<thead>
<tr>
<th>Method</th>
<th>itr</th>
<th>cpu</th>
<th>res</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi</td>
<td>nc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGA</td>
<td>(5, 5, 5)</td>
<td>2.80E-2</td>
<td>(3.63E – 13, 2.47E – 14, 3.57E – 15)</td>
</tr>
<tr>
<td>PCGA</td>
<td>(4, 4, 4)</td>
<td>6.00E-3</td>
<td>(5.35E – 15, 1.05E – 15, 3.40E – 15)</td>
</tr>
</tbody>
</table>

Example 4.2. Here, we consider a random matrix $\tilde{\Psi}$ of order 10 and the vectors $\{\varphi_i\}_{i=1,2,3}$ are chosen to be random vectors of length 10. Fig. 2 shows the respective pseudospectrums of the crisp matrices. From Table 2, we observe that PCGA is more efficient that CGA. In 12 iteration, PCGA approximate $\{\xi_i\}_{i=1,2,3}$ more accurately than CGA. We shall mention that in 1000 iteration the iterative Jacobi, Gauss-Seidel and SOR procedures do not converge.
Fig. 1. Illustration of the pseudospectrums for the crisp matrices of Example 4.1 (a) Φ, (b) Λ and (c) Θ.

Table 2
Results for Example 4.2.

<table>
<thead>
<tr>
<th>Method</th>
<th>CGA</th>
<th>PCGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>itr</td>
<td>(16, 16, 16)</td>
<td>(12, 12, 12)</td>
</tr>
<tr>
<td>cpu</td>
<td>1.00E − 2</td>
<td>9.00E − 3</td>
</tr>
<tr>
<td>res</td>
<td>(1.16E − 10, 1.61E − 11, 1.45E − 10)</td>
<td>(9.58E − 13, 4.46E − 12, 1.24E − 12)</td>
</tr>
</tbody>
</table>

Example 4.3. We consider a random matrix $\tilde{\Psi}$ of order 30 and the vectors $\{\varphi_i\}_{i=1,2,3}$ are chosen to be random vectors of length 30. Fig. 3 shows the respective pseudospectrums of the crisp matrices. We note that for this example the iterative Jacobi, Gauss-Seidel and SOR procedures do not converge in the maximum number of iteration allowed. Table 2 confirms the superiority of PCGA over CGA. For comparison purpose, we find that PCGA requires around three times lesser iterations than CGA for convergence of $\{\xi_i\}_{i=1,2,3}$.

Table 3
Results for Example 4.3.

<table>
<thead>
<tr>
<th>Method</th>
<th>CGA</th>
<th>PCGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>itr</td>
<td>(112, 140, 104)</td>
<td>(31, 42, 30)</td>
</tr>
<tr>
<td>cpu</td>
<td>6.60E − 2</td>
<td>2.70E − 2</td>
</tr>
<tr>
<td>res</td>
<td>(1.43E − 9, 2.53E − 16, 1.24E − 16)</td>
<td>(3.50E − 16, 7.54E − 9, 6.15E − 16)</td>
</tr>
</tbody>
</table>

Example 4.4. Next, we consider a random matrix $\tilde{\Psi}$ of order 50 and the vectors $\{\varphi_i\}_{i=1,2,3}$ are chosen to be random vectors of length 50. Fig. 4 shows the respective pseudospectrums of the crisp matrices. Similar conclusions as in Example 4.3 may be made from Table 4.

Table 4
Results for Example 4.4.

<table>
<thead>
<tr>
<th>Method</th>
<th>CGA</th>
<th>PCGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>itr</td>
<td>(267, 244, 304)</td>
<td>(30, 33, 32)</td>
</tr>
<tr>
<td>cpu</td>
<td>2.19E − 1</td>
<td>3.90E − 2</td>
</tr>
<tr>
<td>res</td>
<td>(4.79E − 7, 1.38E − 16, 2.51E − 16)</td>
<td>(6.78E − 7, 3.51E − 16, 5.87E − 16)</td>
</tr>
</tbody>
</table>
Fig. 2. Illustration of the pseudospectrums for the crisp matrices of Example 4.2 (a) $\Phi$, (b) $\Lambda$ and (c) $\Theta$. 
Fig. 3. Illustration of the pseudospectrums for the crisp matrices of Example 4.3 (a) $\Phi$, (b) $\Lambda$ and (c) $\Theta$. 
Fig. 4. Illustration of the pseudospectrums for the crisp matrices of Example 4.4 (a) $\Phi$, (b) $\Lambda$ and (c) $\Theta$. 
Finally, we consider a manufacturing problem [11]. A manufacturing company has decided to produce three products namely $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$. The company asks his production manager to determine how many number of each product should be produced so that optimum use of the three available machines $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ are made. The available time of the machines that may hinder the process is described in Table 5 and Table 6 depicts the number of machine hours required for each unit of products $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$, respectively.

Table 5
Available machine times.

<table>
<thead>
<tr>
<th>Machine type</th>
<th>Machine hours per month</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{M}_1$</td>
<td>(39.00, 6.10, 7.00)</td>
</tr>
<tr>
<td>$\mathcal{M}_2$</td>
<td>(36.00, 6.80, 5.40)</td>
</tr>
<tr>
<td>$\mathcal{M}_3$</td>
<td>(40.00, 7.10, 9.10)</td>
</tr>
</tbody>
</table>

Table 6
Machine hours required for each unit of the respective product.

<table>
<thead>
<tr>
<th>Machine type</th>
<th>$\mathcal{P}_1$</th>
<th>$\mathcal{P}_2$</th>
<th>$\mathcal{P}_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{M}_1$</td>
<td>(5.00, 0.10, 0.20)</td>
<td>(3.00, 0.30, 0.30)</td>
<td>(4.00, 0.20, 0.10)</td>
</tr>
<tr>
<td>$\mathcal{M}_2$</td>
<td>(3.00, 0.10, 0.30)</td>
<td>(6.00, 0.20, 0.10)</td>
<td>(1.00, 0.40, 0.20)</td>
</tr>
<tr>
<td>$\mathcal{M}_3$</td>
<td>(4.00, 0.20, 0.30)</td>
<td>(1.00, 0.30, 0.10)</td>
<td>(8.00, 0.50, 0.20)</td>
</tr>
</tbody>
</table>

Based on the data provided, the production manager obtains the following FFLS

$$
\begin{pmatrix}
(5.00, 0.10, 0.20) \\
(3.00, 0.10, 0.30) \\
(4.00, 0.20, 0.30)
\end{pmatrix}
\otimes
\begin{pmatrix}
\xi
\end{pmatrix}
=
\begin{pmatrix}
(39.00, 6.10, 7.00) \\
(36.00, 6.80, 5.40) \\
(40.00, 7.10, 9.10)
\end{pmatrix},
$$

where $\xi$ is a fuzzy vector representing the quantity of each product to be produced to optimize the use of the three machines. Table 7 shows the results obtained when solving the above FFLS using CGA and PCGA methods. We observe that the PCGA algorithm converges faster and more accurate as compared to the CGA procedure. Note that the quantity of each product to be produced to satisfy the machines constraints is

$$
\tilde{\xi} =
\begin{pmatrix}
(3.00, 0.20, 0.10) \\
(4.00, 0.60, 0.40) \\
(3.00, 0.30, 0.80)
\end{pmatrix},
$$

and the same is illustrated in Fig. 5.

Table 7
Results for Example 4.5.

<table>
<thead>
<tr>
<th>Method</th>
<th>CGA</th>
<th>PCGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$itr$</td>
<td>(3, 3, 3)</td>
<td>(3, 3, 3)</td>
</tr>
<tr>
<td>$cpu$</td>
<td>$2.03E - 1$</td>
<td>$3.10E - 2$</td>
</tr>
<tr>
<td>$res$</td>
<td>$(0.45E - 14, 0.37E - 14, 0.34E - 14)$</td>
<td>$(1.00E - 16, 0.62E - 16, 0.47E - 16)$</td>
</tr>
</tbody>
</table>
Fig. 5. Illustration of the quantity of $P_1$ (red), $P_2$ (green) and $P_3$ (black) to be produced to satisfy the machines constraints.

5 Conclusion

In this paper, we developed a preconditioned method for finding an approximation of the positive solution of FFLS, where the modal value matrix is positive definite. For comparison purpose, we consider several examples including two applications of FFLS. The numerical results depict that the preconditioned scheme converges to the exact solution more rapidly than the iterative Jacobi, Gauss-Seidel and SOR methods on solving the FFLS.

Acknowledgments: The authors wish to express appreciation to the referees, whose comments improve the original work.

References


http://dx.doi.org/10.2307/2317709


http://dx.doi.org/10.1016/S0019-9958(65)90241-X