International Journal of Mathematical, Engineering and Management Sciences
Vol. 8, No. 1, 60-77, 2023
https://doi.org/10.33889/IJMEMS.2023.8.1.004

Computational Mathematics: Solving Dual Fully Fuzzy Nonlinear Matrix Equations Numerically using Broyden's Method

La Zakaria
Department of Mathematics,
Universitas Lampung, Bandar Lampung, Indonesia.
Corresponding author: lazakaria.1969@fmipa.unila.ac.id

Wahyu Megarani
Department of Mathematics,
Universitas Lampung, Bandar Lampung, Indonesia.
E-mail: wahyumegarani02@gmail.com

Ahmad Faisol
Department of Mathematics,
Universitas Lampung, Bandar Lampung, Indonesia.
E-mail: ahmadfaisol@fmipa.unila.ac.id

Aang Nuryaman
Department of Mathematics,
Universitas Lampung, Bandar Lampung, Indonesia.
E-mail: aang.nuryaman@fmipa.unila.ac.id

Ulfah Muharramah
Business Administration Department,
State Polytechnic of Sriwijaya, Palembang, Indonesia.
E-mail: ulfah.muharramah@polsri.ac.id

(Received on June 03, 2022; Accepted on November 11, 2022)

Abstract
Fuzzy numbers have many applications in various mathematical models in both linear and nonlinear forms. In the form of a nonlinear system, fuzzy nonlinear equations can be constructed in the form of matrix equations. Unfortunately, the matrix equations used to solve the problem of dual fully fuzzy nonlinear systems are still relatively few found in the publication of research results. This article attempts to solve a dual fully fuzzy nonlinear equation system involving triangular fuzzy numbers using Broyden's method. This article provides the pseudocode algorithm and the implementation of the algorithm into the MATLAB program for the iteration process to be carried out quickly and easily. The performance of the given algorithm is the fastest in finding system solutions and provides a minimum error value.

Keywords- Dual fully fuzzy nonlinear system, Matrix equations, Broyden's method, Pseudocode algorithm, Triangular fuzzy numbers.

1. Introduction

Matrix equations can be constructed for many real problems that need to be solved (Lin and Wang, 2013). Further, following developments in mathematics, elements in matrices can now be fuzzy numbers, which can be used to address uncertainty. Fuzzy matrix equations can be solved using analytical and numerical methods. Numerical methods are used when analytical solutions are difficult to obtain. Solutions produced using numerical methods approximate analytical results and have minimal error. Broyden's method,

developed by Broyden (1965), is one of the numerical methods used to solve nonlinear equation systems of and will be used in this article to solve dual fully fuzzy nonlinear matrix equations (DFFNMEs).

There are many authors have written about developing fuzzy analysis, significantly solving fully fuzzy linear and nonlinear systems. Different methods have been used to solve these two types of fully fuzzy systems, including a method for computing the positive solution of a fully fuzzy linear system (Ezzati et al., 2012); a method used to solve a fully fuzzy linear system via decomposing the symmetric coefficient matrix into two equations systems with the Cholesky method (Senthilkumar and Rajendran, 2011); the Jacobi iteration method for solving a fully fuzzy linear system with fuzzy arithmetic (Marzuki, 2015) and triangular fuzzy number (Megarani et al., 2022); a method for finding a positive solution for an arbitrary fully fuzzy linear system with a one-block matrix (Malkawi et al., 2015a); the singular value decomposition method for solving a fully fuzzy linear system (Marzuki et al., 2018); the Gauss–Seidel method for solving a fully fuzzy linear system via alternative multi-playing triangular fuzzy numbers (Deswita and Mashadi, 2019); the Jacobi, Gauss–Seidel, and SOR iterative methods for solving linear fuzzy systems (Inearat and Qatanani, 2018);a linear programming approach utilizing equality constraints to find non-negative fuzzy numbers (Otadi and Mosleh, 2012); combining interval arithmetic with trapezoidal fuzzy numbers to solve a fully fuzzy linear system (Siahlooei and Fazeli, 2018); using an ST decomposition with trapezoidal fuzzy numbers to solve dual fully fuzzy linear systems via alternative fuzzy algebra (Safitri and Mashadi, 2019), using LU factorizations of coefficient matrices for trapezoidal fuzzy numbers to solve dual fully fuzzy linear systems (Marni et al., 2018); combining QR decomposition with trapezoidal fuzzy numbers (Gemawati et al., 2018); and using an ST decomposition with trapezoidal fuzzy numbers to solve a dual fully fuzzy linear system (Jafarian, 2016). In the case of fully fuzzy linear matrix equations, several studies have identified methods for solving them, including a method that utilizes fully fuzzy Sylvester matrix equations (Daud et al., 2018; Elsayed et al., 2022; He et al., 2018; Malkawi et al., 2015b), and a method that finds fuzzy approximate solutions (Guo and Shang, 2013).

Since the beginning of the 21st century, several experts have discussed the problem of fuzzy equation systems with various methods to solve them. Some of them are Daud et al. (2018), Elsayed et al. (2022). Guo and Shang (2013), He et al. (2018), Malkawi et al. (2015b), Ramli et al. (2010), and Jafarian and Jafari (2019). Unfortunately, the presentation of dual fuzzy nonlinear systems has not been widely discussed in matrix equations (See Kołodziejczyk et al., 2020 for numerical methods and Elsayed et al., 2022 for the type of dual fully fuzzy). Research on solving fuzzy nonlinear equations has not been extensive, but methods for solving the equations include Broyden's method (Ramli et al., 2010), and new approaches based on neural networks (Jafari et al., 2020). For FFNMEs, there is a method for finding the non-negative fuzzy number matrix solution that employs a nonlinear system with equality constraints (Jafarian and Jafari, 2019). The fully fuzzy concept can be applied to several fields of research. For example, in the field of uncertain risk assessment, a fuzzy set-based method can be utilized (Ganeshkumar and Arivazhagan, 2016; Qaid and Talbar, 2013; Zeyu et al., 2021). In this article, we propose a numerical procedure to solve the problem of a system of dual fully fuzzy nonlinear equations expressed in a matrix equation with a case study of the problem in the form of a triangular fuzzy set. This article presents an algorithm (pseudocode) and its implementation in a simple MATLAB program for a numerical method (Broyden's method). Based on the output of the program made and the accuracy of the solutions obtained, our proposed method has a high solution performance (single solution in the form of an approximation of the analytical solution with minimal error).

The paper is organized as follows: In section 2, we recall some fundamental results of several concepts of fuzzy number theory, the general form of a fully fuzzy nonlinear matrix equation, and arithmetic for triangular fuzzy numbers. Then, in section 3, we step by step to obtain the solution of the dual fully fuzzy

nonlinear system based on triangular fuzzy numbers using Broyden's method. In section 4, two numerical examples illustrate the proposed algorithm. Finally, in section 5, conclusions are presented.

2. Materials and Methods

Several concepts of fuzzy number theory and FFNME presented in Deswita and Mashadi (2019), Jafarian and Jafari (2019), Jaikumar and Sunantha (2013), Safitri and Mashadi (2019), and Senthilkumar and Rajendran (2011) are reviewed below.

2.1 Fuzzy Set Theory

Definition 1. A fuzzy subset $\tilde{a}$ is defined as $\tilde{a}=(x,\mu_{\tilde{a}}(x))$, where x is a member of set $\tilde{a}$ and $\mu_{\tilde{a}}(x)$ is the membership function with values in the interval $[0,1]$.

Definition 2. Fuzzy numbers are those in a fuzzy set $\tilde{a}:R\to[0,1]$ satisfying the following axioms:

a. $\tilde{a}$ is upper semi-continuous
b. $\tilde{a}(x)=0$ is from the interval $[0,1]$
c. There are real numbers a,b in $[c,d]$ so that
   (i) $\tilde{a}(x)$ is increasing monotonically in $[c,a]$
   (ii) $\tilde{a}(x)$ is decreasing monotonically in $[b,d]$
   (iii) $\tilde{a}(x)=1$ for some x in $[a, b]$

Definition 3. An arbitrary fuzzy number of the form $\tilde{a}=(m-\alpha,m,m-\beta)=(a_m,a_l,a_u)$ with the following membership function:

$$\mu_{\tilde{a}}(x)=\begin{cases}1-\dfrac{m-x}{\alpha}, & m-\alpha\le x\le m,\quad \alpha>0\\ 1-\dfrac{x-m}{\beta}, & m\le x\le m+\beta,\quad \beta>0\\ 0, & \text{otherwise}\end{cases} \tag{1}$$

is called a triangular fuzzy number and has the following graph (see Figure 1):

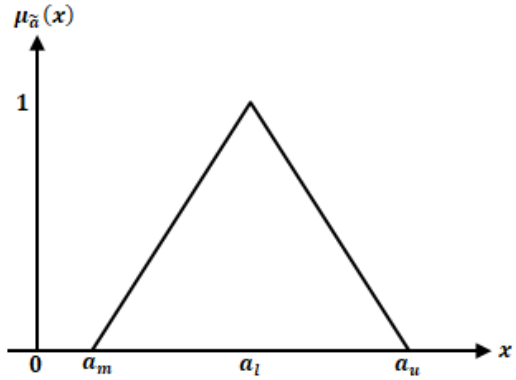

Figure 1. A triangular fuzzy number of the form $\tilde{a}=(a_m,a_l,a_u)$.

Definition 4. A triangular fuzzy number $\tilde{a}=(a_m,a_l,a_u)$ is equal to the triangular fuzzy number $\tilde{b}=(b_m,b_l,b_u)$ if and only if $a_m=b_m$, $a_l=b_l$, and $a_u=b_u$.

Definition 5. A matrix $\tilde{A}=(\tilde{a}_{ij})$ is called a fuzzy matrix if every element of $\tilde{A}$ is a fuzzy number. A fuzzy matrix $\tilde{A}$ is positive (negative), i.e., $\tilde{A}>0$ ($\tilde{A}<0$), if all elements of $\tilde{A}$ are positive (negative). $\tilde{A}$ will be non-positive (non-negative), i.e., $\tilde{A}\leq 0$ ($\tilde{A}\geq 0$), if all elements of $\tilde{A}$ are non-positive (non-negative).

2.2 Triangular Fully Fuzzy Nonlinear Matrix Equations

The following is the general form of a fully fuzzy nonlinear matrix equation (FFNME) with triangular fuzzy numbers (Jafarian and Jafari, 2019):

$$\begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} & \cdots & \tilde{a}_{1n} \\ \tilde{a}_{21} & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{a}_{n1} & \tilde{a}_{n2} & \cdots & \tilde{a}_{nn} \end{bmatrix}\begin{bmatrix} \tilde{x}_{11} \\ \tilde{x}_{21} \\ \vdots \\ \tilde{x}_{n1} \end{bmatrix}+\begin{bmatrix} \tilde{c}_{11} & \tilde{c}_{12} & \cdots & \tilde{c}_{1n} \\ \tilde{c}_{21} & \tilde{c}_{22} & \cdots & \tilde{c}_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{c}_{n1} & \tilde{c}_{n2} & \cdots & \tilde{c}_{nn} \end{bmatrix}\begin{bmatrix} \tilde{x}_{11}^2 \\ \tilde{x}_{21}^2 \\ \vdots \\ \tilde{x}_{n1}^2 \end{bmatrix}+\cdots +\begin{bmatrix} \tilde{e}_{11} & \tilde{e}_{12} & \cdots & \tilde{e}_{1n} \\ \tilde{e}_{21} & \tilde{e}_{22} & \cdots & \tilde{e}_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{e}_{n1} & \tilde{e}_{n2} & \cdots & \tilde{e}_{nn} \end{bmatrix}\begin{bmatrix} \tilde{x}_{11}^n \\ \tilde{x}_{21}^n \\ \vdots \\ \tilde{x}_{n1}^n \end{bmatrix}=\begin{bmatrix} \tilde{b}_{11} \\ \tilde{b}_{21} \\ \vdots \\ \tilde{b}_{n1} \end{bmatrix} \tag{2}$$

where, $\tilde{a}_{ij}$, $\tilde{c}_{ij}$, and $\tilde{e}_{ij}$ for $1\leq i,j\leq n$ are arbitrary triangular fuzzy numbers, and $\tilde{b}_{i1}$ and $\tilde{x}_{i1}$ are nonnegative fuzzy numbers.

2.3 Arithmetic for Triangular Fuzzy Numbers

The arithmetic operations for two triangular fuzzy numbers $\tilde{a}=(a_m,a_l,a_u)$ and $\tilde{b}=(b_m,b_l,b_u)$ are as follows (Jafarian and Jafari, 2019):

(i) $\tilde{a}\oplus\tilde{b}=(a_m+b_m,a_l+b_l,a_u+b_u)$

(ii) $-\tilde{a}=(-a_u,-a_l,-a_m)$

(iii) $\tilde{a}-\tilde{b}=(a_m-b_u,a_l-b_l,a_u-b_m)$

(iv) Multiplication of two triangular fuzzy numbers $\tilde{a}=(a_m,a_l,a_u)$ and $\tilde{b}=(b_m,b_l,b_u)$ is denoted by $\hat{*}$ and based on additional principle but different from classic fuzzy, where $\tilde{a}\hat{*}\tilde{b}=(c_m,c_l,c_u)$ with $c_l=a_l\cdot b_l$, $c_m=\min(a_m\cdot b_m,a_m\cdot b_u,a_u\cdot b_m,a_u\cdot b_u)$, and $c_u=\max(a_m\cdot b_m,a_m\cdot b_u,a_u\cdot b_m,a_u\cdot b_u)$. If $\tilde{a}$ is an arbitrary fuzzy number and $\tilde{b}$ is non-negative, then

$$\tilde{a}\hat{*}\tilde{b}=\begin{cases} (a_m\cdot b_m,a_l\cdot b_l,a_u\cdot b_u), a_m\geq 0, \\ (a_m\cdot b_u,a_l\cdot b_l,a_u\cdot b_u), a_m<0, a_u\geq 0, \\ (a_m\cdot b_m,a_l\cdot b_l,a_u\cdot b_m), a_m<0, a_u<0 \end{cases}. \tag{3}$$

2.4 Broyden's Method

Broyden's method is an extension of the Newton method and a generalization of the Secant method for multiple dimensions. This method first determines an initial value (x_0), then makes its next approximation (x_1) as in the Newton method using the Jacobian matrix

$$J(x)=\begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \cdots & \frac{\partial f_n(x)}{\partial x_n} \end{bmatrix}. \tag{4}$$

The next iteration is determined by replacing the Newton method with the Secant method, as follows:

$$f'(x_n)\approx\frac{f(x_n)-f(x_{n-1})}{x_n-x_{n-1}} \tag{5}$$

In the Broyden's method, the Jacobian matrix $J(x_1)$ is converted to A_1, which is defined as:

$$A_1(x_1-x_0)=F(x_1)-F(x_0) \tag{6}$$

where, (x_1-x_0) is a vector. Any nonzero vector in $\Re^n$ can be written as the sum of multiples of (x_1-x_0).

The orthogonal complement of (x_1-x_0) is specified first to find A_1, as follows:

$$A_1z=J(x_0)z \text{ whenever } (x_1-x_0)^t z=0. \tag{7}$$

Thus, any vector orthogonal to (x_1-x_0) is unaffected by the updated form of $J(x_0)$, and A_1 is defined as:

$$A_1=J(x_0)+\frac{(F(x_1)-F(x_0))-J(x_0)(x_1-x_0)}{(x_1-x_0)^t(x_1-x_0)}(x_1-x_0)^t \tag{8}$$

To find x_2, $J(x_1)$ is replaced by A_1 as follows:

$$x_2=x_1-A_1^{-1}F(x_1) \tag{9}$$

so that the following formulas define the new approximation:

$$\left.\begin{aligned} A_i &= A_{i-1}+\frac{y_i-A_{i-1}s_i}{s_i^t s_i}s_i^t \\ x_{i+1} &= x_i-A_i^{-1}F(x_i) \end{aligned}\right\} \tag{10}$$

where, $y_i=F(x_i)-F(x_{i-1})$ and $s_i=x_i-x_{i-1}$. Using the Sherman-Morrison Theorem (see Ramli et al., 2010), the following invers of A_i is obtained.

$$A_i^{-1} = A_{i-1}^{-1} + \frac{(s_i - A_{i-1}^{-1} y_i) s_i^t A_{i-1}^{-1}}{s_i^t A_{i-1}^{-1} y_i} \tag{11}$$

Nocedal and Wright (2006) noted that the update formula of Broyden's method in Eq. (10) makes the smallest possible change to the Jacobian, and the sequence $\{x_k\}$ generated by Broyden's method in Eq. (10) is well-defined and converges *Q*-superlinearly to x^*. (see Lemma 11.4 and Theorem 11.5 in Nocedal and Wright, 2006).

Any method for solving mathematical equations can be implemented in an algorithm. Sequences, elections, and loops are the three essential structures in any algorithm. Sequence structures are made up of one or more instructions that are executed in the order in which they were written. Election structures include instructions that are executed based on a condition, with the condition being a true or false requirement. If the condition is true, the instruction is executed; if the condition is false, the instruction is not executed. Finally, the same activities are repeated multiple times in looping structures based on a predetermined number or set of circumstances.

Algorithms can be written using descriptive language, flowcharts, or pseudocode. Descriptive sentences provide directions that must be followed in clear language. On the other hand, pseudocode involves a high-level programming language-like approach to writing algorithms. It describes a computer programming algorithm with a simple structure meant only for human comprehension. To be understood by a computer, pseudocode notation must first be translated into the syntax of a certain computer programming language. Finally, flowcharts are a pictorial notation-based method of writing algorithms. A flowchart is a diagram that depicts the sequence of actions in a program as well as the relationships between processes and their statements (Retta et al., 2020).

3. Results

In this section, dual fully fuzzy nonlinear matrix equations (DFFNMEs) are introduced. Broyden's method for solving DFFNMEs is outlined as well.

3.1 Dual Fully Fuzzy Nonlinear Matrix Equations

Consider the following DFFNMEs:

$$\begin{bmatrix} \tilde{a}_{1_{11}} & \tilde{a}_{1_{12}} & \cdots & \tilde{a}_{1_{1n}} \\ \tilde{a}_{1_{21}} & \tilde{a}_{1_{22}} & \cdots & \tilde{a}_{1_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{a}_{1_{n1}} & \tilde{a}_{1_{n2}} & \cdots & \tilde{a}_{1_{nn}} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{bmatrix} + \begin{bmatrix} \tilde{a}_{2_{11}} & \tilde{a}_{2_{12}} & \cdots & \tilde{a}_{2_{1n}} \\ \tilde{a}_{2_{21}} & \tilde{a}_{2_{22}} & \cdots & \tilde{a}_{2_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{a}_{2_{n1}} & \tilde{a}_{2_{n2}} & \cdots & \tilde{a}_{2_{nn}} \end{bmatrix} \begin{bmatrix} \tilde{x}_1^2 \\ \tilde{x}_2^2 \\ \vdots \\ \tilde{x}_n^2 \end{bmatrix} + \cdots$$
$$+ \begin{bmatrix} \tilde{a}_{n_{11}} & \tilde{a}_{n_{12}} & \cdots & \tilde{a}_{n_{1n}} \\ \tilde{a}_{n_{21}} & \tilde{a}_{n_{22}} & \cdots & \tilde{a}_{n_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{a}_{n_{n1}} & \tilde{a}_{n_{n2}} & \cdots & \tilde{a}_{n_{nn}} \end{bmatrix} \begin{bmatrix} \tilde{x}_1^n \\ \tilde{x}_2^n \\ \vdots \\ \tilde{x}_n^n \end{bmatrix} = \begin{bmatrix} \tilde{a}_{n+1_{11}} & \tilde{a}_{n+1_{12}} & \cdots & \tilde{a}_{n+1_{1n}} \\ \tilde{a}_{n+1_{21}} & \tilde{a}_{n+1_{22}} & \cdots & \tilde{a}_{n+1_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{a}_{n+1_{n1}} & \tilde{a}_{n+1_{n2}} & \cdots & \tilde{a}_{n+1_{nn}} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{bmatrix} + \cdots ,$$
$$+ \begin{bmatrix} \tilde{a}_{2n_{11}} & \tilde{a}_{2n_{12}} & \cdots & \tilde{a}_{2n_{1n}} \\ \tilde{a}_{2n_{21}} & \tilde{a}_{2n_{22}} & \cdots & \tilde{a}_{2n_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{a}_{2n_{n1}} & \tilde{a}_{2n_{n2}} & \cdots & \tilde{a}_{2n_{nn}} \end{bmatrix} \begin{bmatrix} \tilde{x}_1^n \\ \tilde{x}_2^n \\ \vdots \\ \tilde{x}_n^n \end{bmatrix} + \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{bmatrix} \tag{12}$$

which can be written in matrix notation as:

$$\tilde{A}_1 \hat{*} \tilde{x} \oplus \tilde{A}_2 \hat{*} \tilde{x}^2 \oplus \ldots \oplus \tilde{A}_n \hat{*} \tilde{x}^n = \tilde{A}_{n+1} \hat{*} \tilde{x} \oplus \tilde{A}_{n+2} \hat{*} \tilde{x}^2 \oplus \ldots \oplus \tilde{A}_{2n} \hat{*} \tilde{x}^n \oplus \tilde{b}, \quad (13)$$

where, $\tilde{A}_i$ (for $1 \leq i \leq 2n$) is a matrix of triangular fuzzy numbers, $\tilde{b}$ and are unknown variables, and $\tilde{x}$ is a column vector of triangular fuzzy numbers. In this case, $\tilde{x}$ is the solution of the DFFNMEs that needs to be found.

3.2 Broyden's Method for Solving DFFNMEs

Broyden's method is constructed to solve nonlinear equation systems and improve Newton's method's performance with respect to storage and Jacobi approximation. In addition, Broyden's method is a generalization of the secant method for multiple dimensions (see Retta et al., 2020) for its application in solving fuzzy nonlinear equations). Broyden's method for solving DFFNMEs can be described as follows:

Step 1. Consider the DFFNMEs in Eq. (13). Transform it into a crisp nonlinear system of equations using triangular fuzzy numbers.

Step 2. By assuming the transformation's result is defined for all positive variables, fix the initial value of $\tilde{x}$ ($x_0 > 0$) as an $n \times 1$ matrix, the tolerance (*tol*), and the maximum number of iterations (*N*).

Step 3. Find the Jacobian matrix and substitute the initial value into it to arrive at:

$$A_0 = J(x_0)$$

Step 4. Compute the first iteration using the Newton method, i.e.,

$$x_1 = x_0 - A_0^{-1} F(x).$$

Step 5. For $i \geq 1$, update the Jacobian matrix using $y_i = F(x_i) - F(x_{i-1})$ and $s_i = x_i - x_{i-1}$.

Step 6. Update the inverse of the Jacobian matrix using the following formula:

$$A_i^{-1} = A_{i-1}^{-1} + \frac{(s_i - A_{i-1}^{-1} y_i) s_i^t A_{i-1}^{-1}}{s_i^t A_{i-1}^{-1} y_i}$$

Step 7. Compute the next approximation using the following formula:

$$x_{i+1} = x_i - A_i^{-1} F(x_i)$$

Step 8. For $1 \leq i \leq N$, $N =$ maximum number of iterations, repeat steps 5 to 7 until the error ε less than or equal to the tolerance *tol*, where the error ε is

$$\varepsilon = \left\| x_{i+1} - x_i \right\|$$

3.3 Flowchart for Solving DFFNMEs Using Broyden's Method

View step by step Broyden's method for solving DFFNMEs as described in Subsection 3.2. In flowchart form, the visualization of the algorithm for solving DFFNMEs using Broyden's method is given in Figure 2.

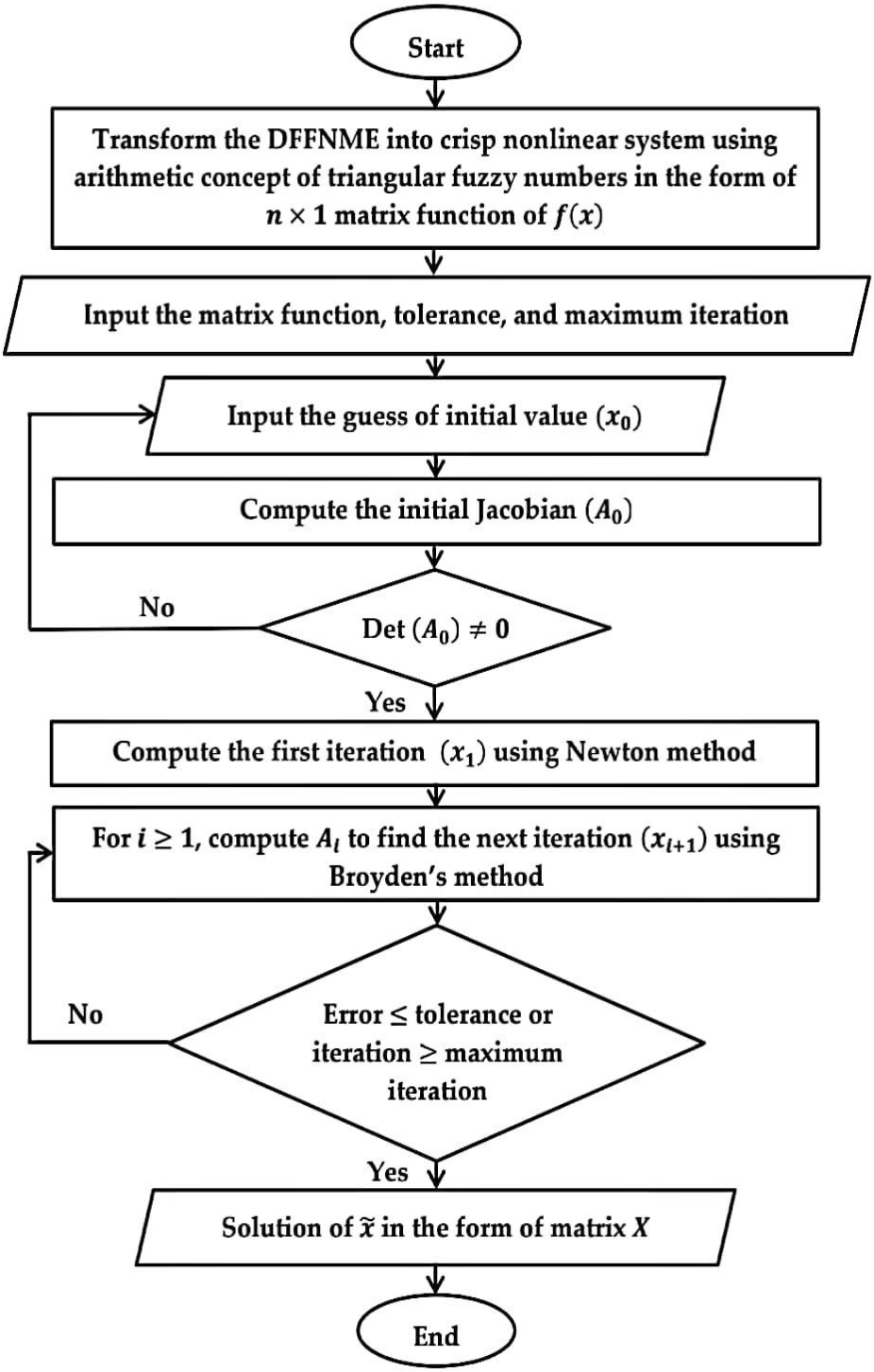

Figure 2. Flowchart for solving DFFNMEs using Broyden's method.

3.4 Pseudocode for Broyden's Algorithm

The pseudocode for Broyden's method to solve the DFFNME system converted into a crisp nonlinear system is shown in Algorithm 3.4.1 below.

Algorithm 3.4.1

{Start condition: A nonlinear system is defined. Parameter values such as the tolerance, maximum number of iterations, and an initial guess x_0 are fixed.
End condition: The system's solution is found using Broyden's method.}
Functionalgorithm_Broyden(f,x: array[1..6] of real; A0,A: array[1..6,1..6] of real; x_1,s,y,F,F_1: array[1..2] of real; st,ut: array[1..1,1..6] of real; X: array[1..k,1..6] of real;eps,p: real; k,Nmax: integer}

```
Read
(f,x,eps,k,Nmax)
begin
jac ← jacobian(f,[x1 x2 x3 y1 y2 y3],x);
A0 ← subs(jac,[x1;x2;x3;y1;y2;y3],x);
while det(A0) not equal to zero do
A ← double(inv(A0));
F ← double(subs(f,[x1;x2;x3;y1;y2;y3],x));
s ←-A*F;
x_1←x+s;
X← [x';x_1'];
x ←x_1;
        while (err>eps)&&k<Nmax
        begin
        F_1←F;
        F←double(subs(f,[x1;x2;x3;y1;y2;y3],x));
        y ←double(F-F_1);
        st←transpose(s);
        p←-st*(-A*y);
        ut←st*A;
        A←A+(1/p)*(s-A*y)*ut;
        s ←-A*F;
        x_new←x+s;
        X←[X;x_new'];
        err←norm(x_new-x);
        x←x_new;
        k←k+1;
        endwhile
write(X)
endwhile
endfunction
```

{loop end condition : k >Nmax& err <eps, so the solutions are the array of X}

3.5 Implementation of Broyden's Algorithm Using MATLAB Programming

The implementation of Broyden's algorithm proceeds in MATLAB programming as follows.

MATLAB Program for Algorithm 3.4.1

```
functionalgorithm_Broyden
clear all;
clc;
%Function to solve a dual fully fuzzy nonlinear matrix equation using Broyden's method
syms x1 x2 x3 y1 y2 y3
f=[ ]; %Transformation of dual fully fuzzy nonlinear matrix equation into crisp nonlinear system
jac=jacobian(f,[x1 x2 x3 y1 y2 y3]);
x=[ ]; %Initial guess value
A0=subs(jac,[x1;x2;x3;y1;y2;y3],x); %Initial Jacobian matrix
if det(A0)~=0
A=double(inv(A0));
F=double(subs(f,[x1;x2;x3;y1;y2;y3],x));

%Step 1 to find first iteration using Newton method
%Syntax: x1=x0-A*F(x)
s=-A*F;
x_1=x+s;
X=[x';x_1'];
x=x_1; %Update the approximation
k=1;
err=norm(x); %Error of iteration
eps= ; %Tolerance
Nmax= ; %Maximum iteration

%Step 2 to find next iteration using Broyden method
while (err>eps)&&k<Nmax
   F_1=F;
   F=double(subs(f,[x1;x2;x3;y1;y2;y3],x));
   y=double(F-F_1);
   st=transpose(s);
   p=-st*(-A*y);
   ut=st*A;
   A=A+(1/p)*(s-A*y)*ut;
   s=-A*F;
   x_new=x+s;
   X=[X;x_new'];
   err=norm(x_new-x);
   x=x_new;
   k=k+1;
end
X
else
   disp('The initial Jacobian matrix is not nonsingular. Re-input another initial guess value.')
end
End
```

3.6 Comparison with Other Methods

In this section, we provide information about several methods of solving fuzzy equation problems published by several researchers to see the novelty of the algorithms provided by the authors in this article. Ramli et al. (2010) introduced the algorithm of Broyden's method to solve a fuzzy nonlinear equation that puts the equation in parametric form. The method is limited to solving one nonlinear equation and does not involve triangular fuzzy numbers. Thus, it is different from our proposed algorithm. Jafarian (2016) and Jafari et al. (2020) introduced a method for solving FFMNEs (Jafari et al., 2020; Jafarian, 2016). This method eliminates the artificial variables in nonlinear programming without using an algorithm. So, it can take a longer to solve DFFNMEs. Based on this information, this paper extended the method with a numerical method and algorithm, allowing the optimal solution to be found relatively quickly. Elsayed et al. (2022) introduced an algorithm to solve a fully fuzzy Sylvester linear matrix equation. However, this paper considers a nonlinear matrix equation that is dual fully fuzzy. Finally, Sulaiman et al. (2018) introduced the Shamanskii method for solving a single dual fully fuzzy nonlinear equation, not a nonlinear system.

4. Numerical Example

In this section, two numerical examples are given to illustrate the proposed algorithm.

Example 4.1 Consider the following DFFNMEs:

$$\begin{bmatrix}(2,3,5) & (2,4,5)\\ (1,2,3) & (3,4,6)\end{bmatrix}\begin{bmatrix}\tilde{x}\\ \tilde{y}\end{bmatrix}+\begin{bmatrix}(1,2,3) & (3,5,6)\\ (3,4,5) & (1,3,4)\end{bmatrix}\begin{bmatrix}\tilde{x}^2\\ \tilde{y}^2\end{bmatrix}=\begin{bmatrix}(2,4,5) & (3,4,5)\\ (2,3,4) & (2,4,5)\end{bmatrix}\begin{bmatrix}\tilde{x}^2\\ \tilde{y}^2\end{bmatrix}+\begin{bmatrix}(5,12,42)\\ (4,24,47)\end{bmatrix}, \tag{14}$$

where, $\tilde{x}$ and $\tilde{y}$ are triangular fuzzy numbers. With a tolerance of 10^{-8}, a maximum number of iterations of 100, and initial values $\tilde{x}=(1,2,3)$ and $\tilde{y}=(1,2,3)$, the solution of the system can be found as follows:

Assume $\tilde{x}=(x_1,x_2,x_3)$ and $\tilde{y}=(y_1,y_2,y_3)$. Then the given DFFNMEs can be written as

$$\begin{cases}(2,3,5)\hat{*}(x_1,x_2,x_3)\oplus(2,4,5)\hat{*}(y_1,y_2,y_3)\oplus(1,2,3)\hat{*}(x_1^2,x_2^2,x_3^2)\\ \oplus(3,5,6)\hat{*}(y_1^2,y_2^2,y_3^2)=(2,4,5)\hat{*}(x_1^2,x_2^2,x_3^2)\oplus(3,4,5)\hat{*}(y_1^2,y_2^2,y_3^2)\oplus(5,12,42)\\ \\ (1,2,3)\hat{*}(x_1,x_2,x_3)\oplus(3,4,6)\hat{*}(y_1,y_2,y_3)\oplus(3,4,5)\hat{*}(x_1^2,x_2^2,x_3^2)\\ \oplus(1,3,4)\hat{*}(y_1^2,y_2^2,y_3^2)=(2,3,4)\hat{*}(x_1^2,x_2^2,x_3^2)\oplus(2,4,5)\hat{*}(y_1^2,y_2^2,y_3^2)\oplus(2,24,47)\end{cases} \tag{15}$$

The system in Eq. (15) can be also transformed into a crisp nonlinear system using the triangular fuzzy numbers:

$$\begin{cases}2x_1+2y_1+x_1^2+3y_1^2=2x_1^2+3y_1^2+5\\ 3x_2+4y_2+2x_2^2+5y_2^2=4x_2^2+4y_2^2+12\\ 5x_3+5y_3+3x_3^2+6y_3^2=5x_3^2+5y_3^2+42\\ x_1+3y_1+3x_1^2+y_1^2=2x_1^2+2y_1^2+4\\ 2x_2+4y_2+4x_2^2+3y_2^2=3x_2^2+4y_2^2+24\\ 3x_3+6y_3+5x_3^2+4y_3^2=4x_3^2+5y_3^2+47\end{cases} \tag{16}$$

By applying crisp arithmetic operations, the nonlinear system in Eq. (16) can be simplified to:

$$\begin{cases} 2x_1+2y_1-x_1^2=5 \\ 3x_2+4y_2-2x_2^2+y_2^2=12 \\ 5x_3+5y_3-2x_3^2+y_3^2=42 \\ x_1+3y_1+x_1^2-y_1^2=4 \\ 2x_2+4y_2+x_2^2-y_2^2=24 \\ 3x_3+6y_3+x_3^2-y_3^2=47 \end{cases} \tag{17}$$

Next, the system in Eq. (17) will be solved via the algorithm of Broyden's method using MATLAB programming. The solution is provided in Table 1.

Table 1. Solutions for x_1 to x_3 and y_1 to y_3 for Example 4.1.

Iteration	x_1	x_2	x_3	y_1	y_2	y_3
0	1.0000	2.0000	3.0000	1.0000	2.0000	3.0000
1	0.6667	4.0000	5.2222	2.0000	3.5000	6.3232
2	0.9819	3.6690	5.9921	2.0630	4.1088	6.6927
3	1.0439	4.1329	5.8295	2.0163	3.9031	6.9841
⋮	⋮	⋮	⋮	⋮	⋮	⋮
14	1.0000	4.0000	6.0000	2.0000	4.0000	7.0000

Table 1 shows that the iteration process ends in the fourteenth iteration with the solution $\tilde{x}=(1,4,6)$ and $\tilde{y}=(2,4,7)$ after a total running time of 2.81 seconds. The solution can be represented by the membership function in Table 2 and shown by the diagrams in Figure 3.

Table 2. The member function for $0\le\mu\le1$ and Example 4.1.

α	$x(\mu)$		$y(\mu)$	
	$\underline{x}(\mu)$	$\overline{x}(\mu)$	$\underline{y}(\mu)$	$\overline{y}(\mu)$
0	1.0000	6.0000	2.0000	7.0000
0.1	1.3000	5.8000	2.2000	6.7000
0.2	1.6000	5.6000	2.4000	6.4000
0.3	1.9000	5.4000	2.6000	6.1000
0.4	2.2000	5.2000	2.8000	5.8000
0.5	2.5000	5.0000	3.0000	5.5000
0.6	2.8000	4.8000	3.2000	5.2000
0.7	3.1000	4.6000	3.4000	4.9000
0.8	3.4000	4.4000	3,6000	4.6000
0.9	3.7000	4.2000	3.8000	4.3000
1	4.0000	4.0000	4.0000	4.0000

In Figure 3, two diagrams each show that solution $\tilde{x}$ is a triangular fuzzy number with a minimum value equal to 1, an average value equal to 4, and a maximum value equal to 6 (see Figure 3 in the left side). Meanwhile, Figure 3 in the right side, the $\tilde{y}$ solution is a triangular fuzzy number with a minimum value equal to 2, an average value equal to 4, and a maximum value equal to 7.

The algorithm's performance can be measured through the processing time or the number of iterations required by the algorithm. Some initial guess values are used in the algorithm to solve the given problem (Eq. 17). For example, for the three initial guess values given (see Table 3, second column), the proposed

algorithm provides the iteration with measurement results (column fourth) and processing time (column fifth).

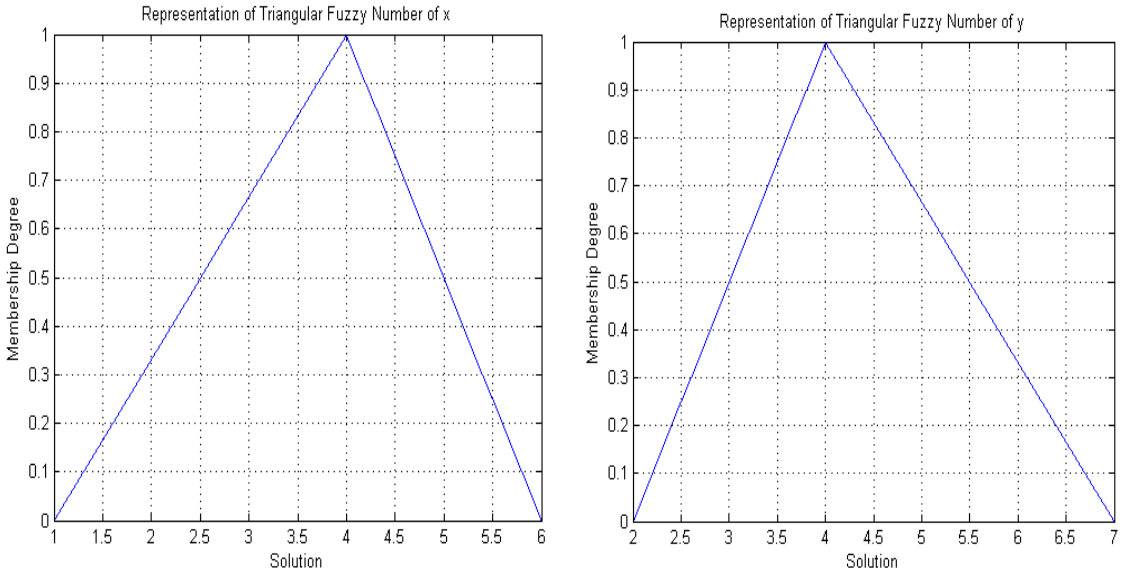

Figure 3. Representations of the triangular fuzzy numbers $\tilde{x}$ (left) and $\tilde{y}$ (right) for Example 4.1.

Table 3. Comparison of the performance (number of iterations and running time) of the method to find a solution of Example 4.1 based on three different initial guess values of $\tilde{x}_0$.

No	Initial Guess Values	Solution	Number of Iterations	Running Time (in second)
1	$\tilde{x}_0$ = (0.5, 1, 1.5)	$\tilde{x}$ = (1, 4, 6)	17	2.87
	$\tilde{y}_0$ = (1, 1.5 2)	$\tilde{y}$ = (2, 4, 7)		
2	$\tilde{x}_0$ = (1, 2, 3)	$\tilde{x}$ = (1, 4, 6)	14	2.81
	$\tilde{y}_0$ = (1, 2, 3)	$\tilde{y}$ = (2, 4, 7)		
3	$\tilde{x}_0$ = (1.5, 5, 6)	$\tilde{x}$ = (1, 4, 6)	12	2.68
	$\tilde{y}_0$ = (3, 6, 7)	$\tilde{y}$ = (2, 4, 7)		

Table 3 shows that by taking different initial guess values, the system solved by Broyden's method (MATLAB Program for Algorithm 3.4.1) produces a solution that converges to the exact value.

Example 4.2 In this example, a chemistry process used as an example in Jafari and Yu (2015) is modified into a DFFNME system. To make a desired substance (DS), a chemical reaction is used to combine poly ethylene (PE) with poly propylene (PP). The cost of PE is $\tilde{x}$, and the cost of PP is $\tilde{y}$. PE and PP have unknown weights, yet they satisfy the triangle function. Two different types of DS under two different sets of conditions are needed. What are $\tilde{x}$ and $\tilde{y}$ to total $F(4.6,6.2,7.6)=z_1^*$ and $F(4.5,5.6,6.7)=z_2^*$ under the two

sets of conditions? The relationships between the weights and costs are shown in the following DFFNME system:

$$\begin{bmatrix} (2.5,3,3.25) & (2,2.25,2.75) \\ (2,2.5,2.75) & (1.5,2,2.25) \end{bmatrix}\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}+\begin{bmatrix} (1.75,2,2.5) & (1.25,2,2.25) \\ (1.5,1.75,2) & (1,1.5,2) \end{bmatrix}\begin{bmatrix} \tilde{x}^2 \\ \tilde{y}^2 \end{bmatrix} = \begin{bmatrix} (0.75,1,1.25) & (1.75,2,2.5) \\ (1,1.5,1.75) & (0.5,1,1.25) \end{bmatrix}\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}+\begin{bmatrix} (1.5,1.75,2.25) & (1,1.75,2) \\ (1.25,1.5,1.75) & (0.75,1.25,1.75) \end{bmatrix}\begin{bmatrix} \tilde{x}^2 \\ \tilde{y}^2 \end{bmatrix}+\begin{bmatrix} (4.6,6.2,7.6) \\ (4.5,5.6,6.7) \end{bmatrix} \tag{18}$$

Consider the MATLAB Program for Algorithm 3.4.1. For example, to solve $\tilde{x}$ and $\tilde{y}$ in Eq. (18), given a tolerance of 10^{-8}, a maximum number of iterations of 100, and initial values $\tilde{x}=(1,2,3)$ and $\tilde{y}=(1,2,3)$, the solution of the DFFNME system can be obtained as follows:

Assume $\tilde{x}=(x_1,x_2,x_3)$ and $\tilde{y}=(y_1,y_2,y_3)$. Then the given DFFNME system can be written as

$$\begin{cases} (2.5,3,3.25)\hat{*}(x_1,x_2,x_3)\oplus(2,2.25,2.75)\hat{*}(y_1,y_2,y_3)\oplus(1.75,2,2.5)\hat{*}(x_1^2,x_2^2,x_3^2) \\ \oplus(1.25,2,2.25)\hat{*}(y_1^2,y_2^2,y_3^2)=(0.75,1,1.25)\hat{*}(x_1,x_2,x_3)\oplus(1.75,2,2.5)\hat{*}(y_1,y_2,y_3) \\ \oplus(1.5,1.75,2.25)\hat{*}(x_1^2,x_2^2,x_3^2)\oplus(1,1.75,2)\hat{*}(y_1^2,y_2^2,y_3^2)\oplus(4.6,6.2,7.6) \\ \\ (2,2.5,2.75)\hat{*}(x_1,x_2,x_3)\oplus(1.5,2,2.25)\hat{*}(y_1,y_2,y_3)\oplus(1.5,1.75,2)\hat{*}(x_1^2,x_2^2,x_3^2) \\ \oplus(1,1.5,2)\hat{*}(y_1^2,y_2^2,y_3^2)=(1,1.5,1.75)\hat{*}(x_1,x_2,x_3)\oplus(0.5,1,1.25)\hat{*}(y_1,y_2,y_3) \\ \oplus(1.25,1.5,1.75)\hat{*}(x_1^2,x_2^2,x_3^2)\oplus(0.75,1.25,1.75)\hat{*}(y_1^2,y_2^2,y_3^2)\oplus(4.5,5.6,6.7) \end{cases} \tag{19}$$

The system in Eq. (19) can be transformed into a crisp nonlinear system using triangular fuzzy numbers:

$$\begin{cases} 2.5x_1+2y_1+1.75x_1^2+1.25y_1^2=0.75x_1+1.75y_1+1.5x_1^2+y_1^2+4.6 \\ 3x_2+2.25y_2+2x_2^2+2y_2^2=x_2+2y_2+1.75x_2^2+1.75y_2^2+6.2 \\ 3.25x_3+2.75y_3+2.5x_3^2+2.25y_3^2=1.25x_3+2.5y_3+2.25x_3^2+2y_3^2+7.6 \\ 2x_1+1.5y_1+1.5x_1^2+y_1^2=x_1+0.5y_1+1.25x_1^2+0.75y_1^2+4.5 \\ 2.5x_2+2y_2+1.75x_2^2+1.5y_2^2=1.5x_2+y_2+1.5x_2^2+1.25y_2^2+5.6 \\ 2.75x_3+2.25y_3+2x_3^2+2y_3^2=1.75x_3+1.25y_3+1.75x_3^2+1.75y_3^2+6.7 \end{cases} \tag{20}$$

By applying crisp arithmetic operations, the nonlinear system in Eq. (20) can be simplified to:

$$\begin{cases} 1.75x_1+0.25y_1+0.25x_1^2+0.25y_1^2=4.6 \\ 2x_2+0.25y_2+0.25x_2^2+0.25y_2^2=6.2 \\ 2x_3+0.25y_3+0.25x_3^2+0.25y_3^2=7.6 \\ x_1+y_1+0.25x_1^2+0.25y_1^2=4.5 \\ x_2+y_2+0.25x_2^2+0.25y_2^2=5.6 \\ x_3+y_3+0.25x_3^2+0.25y_3^2=6.7 \end{cases} \tag{21}$$

Next, by applying MATLAB Program for Algorithm 3.4.1 in § 3.5, the nonlinear system in Eq. (21) can be solved (see Table 4 for their solution).

Table 4 shows that the iteration process ends in the seventh iteration with the solution $\tilde{x}=(1.676,1.9698,2.3711)$ and $\tilde{y}=(1.5383,1.8264,1.9615)$ after a total running time of 2.4 seconds. The solution can be represented by the membership function in Table 5 and shown by the diagrams in Figure 4.

Table 4. Solutions for x_1 to x_3 and y_1 to y_3 for Example 4.2.

Iteration	x_1	x_2	x_3	y_1	y_2	y_3
0	1.0000	2.0000	3.0000	1.0000	2.0000	3.0000
1	1.7333	1.9714	2.4343	1.6000	1.8286	2.0457
2	1.6586	1.9698	2.3816	1.5253	1.8264	1.9755
3	1.6724	1.9697	2.3724	1.5391	1.8263	1.9632
⋮	⋮	⋮	⋮	⋮	⋮	⋮
7	1.6716	1.9698	2.3711	1.5383	1.8264	1.9615

Table 5. The member function for $0 \le \mu \le 1$ and Example 4.2.

α	$x(\mu)$		$y(\mu)$	
	$\underline{x}(\mu)$	$\overline{x}(\mu)$	$\underline{y}(\mu)$	$\overline{y}(\mu)$
0	1.67160156	2.37111381	1.53826822	1.96148508
0.1	1.70141782	2.33097885	1.56707663	1.94797180
0.2	1.73123409	2.29084389	1.59588504	1.93445853
0.3	1.76105036	2.25070894	1.62469345	1.92094525
0.4	1.79086663	2.21057398	1.65350186	1.90743197
0.5	1.82068289	2.17043902	1.68231026	1.89391869
0.6	1.85049916	2.13030406	1.71111867	1.88040542
0.7	1.88031543	2.09016910	1.73992708	1.86689214
0.8	1.91013169	2.05003415	1.76873549	1.85337886
0.9	1.93994796	2.00989919	1.79754390	1.83986558
1	1.96976423	1.96976423	1.82635231	1.82635231

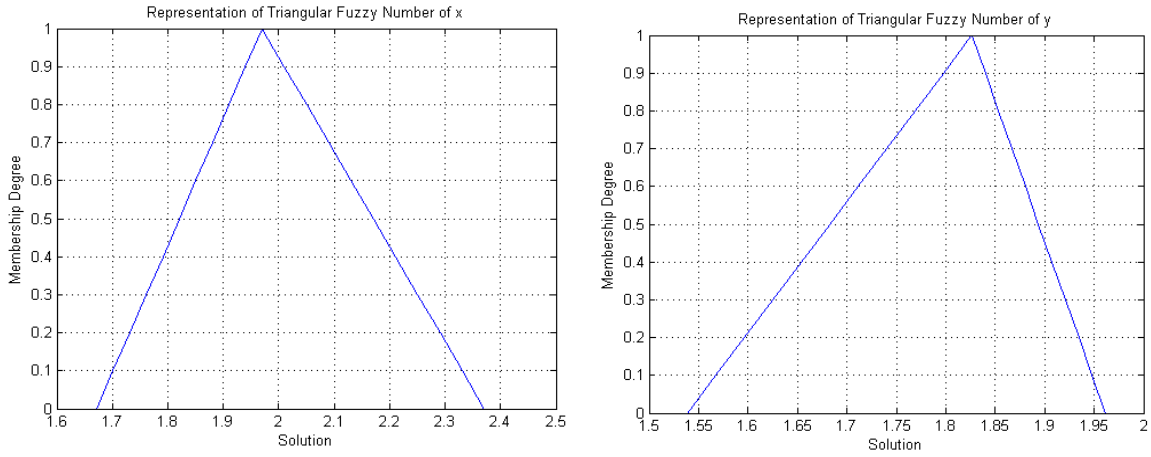

Figure 4. Representations of the triangular fuzzy numbers $\tilde{x}$ (left) and $\tilde{y}$ (right) for Example 4.2.

Figure 4 shows the cost of PE and PP needed to make a desired substance. Figure 4 in the left side shows that the minimum cost of PE required is 1.67, the average cost is 1.97, and the maximum cost is 2.37. While Figure 4 in the right side shows that the minimum cost of PP required is 1.54, the average cost is 1.83, and the maximum cost is 1.96.

Some initial guess values are used in the algorithm to solve problem in Eq. (21). For example, for the three initial guess values given (see Table 6, second column), the proposed algorithm provides the iteration with measurement results (column fourth) and processing time (column fifth).

Table 6. Comparison of the performance (number of iterations and running time) of the method to find a solution of Example 4.2 based on three different initial guess values of $\tilde{x}_0$.

No	Initial Guess Values	Solution	Number of Iterations	Running Time (in second)
1	$\tilde{x}_0$ = (0.5, 1, 1.5)	$\tilde{\tilde{x}}$ = (1.6716, 1.9698, 2.3711)	9	2.457
	$\tilde{y}_0$ = (0.5, 0.75, 1)	$\tilde{y}$ = (1.5383, 1.8264, 1.9615)		
2	$\tilde{x}_0$ = (1, 2, 3)	$\tilde{\tilde{x}}$ = (1.6716, 1.9698, 2.3711)	7	2.4
	$\tilde{y}_0$ = (1, 2, 3)	$\tilde{y}$ = (1.5383, 1.8264, 1.9615)		
3	$\tilde{x}_0$ = (5, 5.5, 5.75)	$\tilde{\tilde{x}}$ = (1.6716, 1.9698, 2.3711)	12	2.749
	$\tilde{y}_0$ = (6.25, 6.5, 7)	$\tilde{y}$ = (1.5383, 1.8264, 1.9615)		

Table 6 shows that by taking different initial guess values, the system solved by Broyden's method (MATLAB Program for Algorithm 3.4.1) produces a solution that converges to the same value for four-digits precision.

5. Conclusions

This paper has shown that DFFNMEs can be written as a matrix equation and solved numerically using Broyden's method through our proposed algorithm and programming. Two examples of DFFNME systems have been solved using the proposed algorithm and provide effective performance results in determining the solution to each problem. Furthermore, the number of iterations is less than 20, and the problem-solving process time is less than 3 seconds, which are perfect indicators of the performance of this proposed algorithm. Note that further research opportunities can be developed from this article. Such as the use of fuzzy sets other than fuzzy triangular numbers (see (Kumar et al., 2021)) for a hesitant fuzzy set discussion) and the application of a proposed algorithm for solving practical problems involving the Fuzzy Transportation Problem (see (Muthuperumal et al., 2020)). Like the Fuzzy Transportation Problem, the reader can modify the proposed algorithm to solve the practical problem involving the Traveling Salesman Problem, Assignment Problem, Scheduling Problem, and Critical Path Methods Problem.

Conflict of Interest

The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgments

The authors would like to thank the Head of the Institute of Research and Community Services of Lampung University, who supported and funded this research.

References

Broyden, C.G. (1965). A class of methods for solving nonlinear simultaneous questions. *Mathematics of Computation*, *19*(92), 577-593. https://doi.org/10.2307/2003941.

Daud, W.S.W., Ahmad, N., & Malkawi, G. (2018). Solving arbitrary fully fuzzy Sylvester matrix equations and its theoretical foundation. In *AIP Conference Proceedings* (Vol. 2013, No. 1, p. 020026). AIP Publishing LLC. https://doi.org/10.1063/1.5054225.

Desmita, Z., & Mashadi, M. (2019). Alternative multiplying triangular fuzzy number and applied in fully fuzzy linear system. *American Academic Scientific Research Journal for Engineering, Technology, and Sciences*, *56*(1), 113-123.

Elsayed, A.A.A., Saassouh, B., Ahmad, N., & Malkawi, G. (2022). Two-stage algorithm for solving arbitrary trapezoidal fully fuzzy Sylvester matrix equations. *Symmetry*, *14*(3), 446. https://doi.org/10.3390/sym14030446.

Ezzati, R., Khezerloo, S., & Yousefzadeh, A. (2012). Solving fully fuzzy linear system of equations in general form. *Journal of Fuzzy Set Valued Analysis*, *2012*, 1-11. https://doi.org/10.5899/2012/jfsva-00117.

Ganeshkumar, K., & Arivazhagan, D. (2016). New cryptography algorithm with fuzzy logic for effective data communication. *Indian Journal of Science and Technology*, *9*(48), 1-6. https://doi.org/10.17485/ijst/2016/v9i48/108970.

Gemawati, S., Nasfianti, I., & Hadi, A. (2018). A new method for dual fully fuzzy linear system with trapezoidal fuzzy numbers by QR decomposition. In *Journal of Physics: Conference Series* (Vol. 1116, No. 2, p. 022011). IOP Publishing. https://doi.org/10.1088/1742-6596/1116/2/022011.

Guo, X., & Shang, D. (2013). Fuzzy approximate solution of positive fully fuzzy linear matrix equations. *Journal of Applied Mathematics*, *2013*. Article ID 178209. https://doi.org/10.1155/2013/178209.

He, Q., Hou, L., & Zhou, J. (2018). The solution of fuzzy Sylvester matrix equation. *Soft Computing*, *22*(19), 6515-6523. https://doi.org/10.1007/s00500-017-2702-8.

Inearat, L., & Qatanani, N. (2018). Numerical methods for solving fuzzy linear systems. *Mathematics*, *6*(2), 19. https://doi.org/10.3390/math6020019.

Jafari, R., Razvarz, S., & Gegov, A. (2020). A novel technique for solving fully fuzzy nonlinear systems based on neural networks. *Vietnam Journal of Computer Science*, *7*(1), 93-107. https://doi.org/10.1142/S2196888820500050.

Jafari, R., & Yu, W. (2015). Fuzzy control for uncertainty nonlinear systems with dual fuzzy equations. *Journal of Intelligent and Fuzzy Systems*, *29*(3), 1229-1240. https://doi.org/10.3233/IFS-151731.

Jafarian, A. (2016). New decomposition method for solving dual fully fuzzy linear systems. *International Journal of Fuzzy Computation and Modelling*, *2*(1), 76-85. https://doi.org/10.1504/ijfcm.2016.077883.

Jafarian, A., & Jafari, R. (2019). A new computational method for solving fully fuzzy nonlinear matrix equations. *International Journal of Fuzzy Computation and Modelling*, *2*(4), 275-285. https://doi.org/10.1504/IJFCM.2019.100317.

Jaikumar, K., & Sunantha, S. (2013). S S T decomposition method for solving fully fuzzy linear systems. *International Journal of Industrial Mathematics*, *5*(4), 275-280.

Kołodziejczyk, J., Piegat, A., & Sałabun, W. (2020). Which alternative for solving dual fuzzy nonlinear equations is more precise? *Mathematics*, *8*(9), (1507). https://doi.org/10.3390/math8091507.

Kumar, A., Bisht, S., Goyal, N., & Ram, M. (2021). Fuzzy reliability based on hesitant and dual hesitant fuzzy set evaluation. *International Journal of Mathematical, Engineering and Management Sciences*, *6*(1), 166-179. https://doi.org/10.33889/IJMEMS.2021.6.1.010.

Lin, Y., & Wang, Q.W. (2013). Iterative solution to a system of matrix equations. *Abstract and Applied Analysis*, *2013*. Article ID 124979. https://doi.org/10.1155/2013/124979.

Malkawi, G., Ahmad, N., & Ibrahim, H. (2015a). An algorithm for a positive solution of arbitrary fully fuzzy linear system. *Computational Mathematics and Modeling*, *26*(3), 436-465. https://doi.org/10.1007/s10598-015-9283-0.

Malkawi, G., Ahmad, N., & Ibrahim, H. (2015b). Solving the fully fuzzy sylvester matrix equation with triangular fuzzy number. *Far East Journal of Mathematical Sciences*, *98*(1), 37-55. https://doi.org/10.17654/FJMSSep2015_037_055.

Marni, S.I., Mashadi, & Gemawati, S. (2018). Solving dual fully fuzzy linear system by use factorizations of the coefficient matrix for trapezoidal fuzzy number. *Bulletin of Mathematics*, *10*(02), 145-156.

Marzuki, C.C. (2015). Penyelesaian sistem persamaan linear fully fuzzy menggunakan metode iterasi jacobi. *Journal Sains, Teknologi Dan Statistika*, *1*(1), 1-7. http://dx.doi.org/10.24014/jsms.v1i1.1965.

Marzuki, C.C., Agustian`, A., Hariati, D., Afmilda, J., Husna, N., & Nanda, P. (2018). Penyelesaian Sistem persamaan linier fully fuzzy menggunakan metode dekomposisi nilai singular (SVD). *Jurnal Matematika "MANTIK," 4*(2), 143-149. https://doi.org/10.15642/mantik.2018.4.2.143-149.

Megarani, W., Zakaria, L., Sutrisno, A., Aziz, D., & Nuryaman, A. (2022). Algorithms and programming: The jacobi method for solving dual fully fuzzy linear systems. *Recent Advances in Computer Science and Communications*, Advance online publication. https://doi.org/10.2174/2666255815666220511123035.

Muthuperumal, S., Titus, P., & Venkatachalapathy, M. (2020). An algorithmic approach to solve unbalanced triangular fuzzy transportation problems. *Soft Computing*, *24*, 18689-18698. https://doi.org/10.1007/s00500-020-05103-3.

Nocedal, J., & Wright, S.J. (2006). *Numerical optimization*. 2nd ed. Springer Science+Business Media, LLC.

Otadi, M., & Mosleh, M. (2012). Solving fully fuzzy matrix equations. *Applied Mathematical Modelling*, *36*(12), 6114-6121. https://doi.org/10.1016/j.apm.2012.02.005.

Qaid, G.R.S., & Talbar, S.N. (2013). Encrypting image by using fuzzy logic algorithm. *International Journal of Image Processing and Vision Science*, *2*(1), 25-29. https://doi.org/10.47893/ijipvs.2013.1059.

Ramli, A., Abdullah, M.L., & Mamat, M. (2010). Broyden's method for solving fuzzy nonlinear equations. *Advances in Fuzzy Systems*, *2010*. Article ID 763270. https://doi.org/10.1155/2010/763270.

Retta, A.M., Isroqmi, A., & Nopriyanti, T.D. (2020). Pengaruh penerapan algoritma terhadap pembelajaran pemrograman komputer. *Indiktika : Jurnal Inovasi Pendidikan Matematika*, *2*(2), 126-135.

Safitri, Y., & Mashadi. (2019). Alternative fuzzy algebra to solve dual fully fuzzy linear system using st decomposition method. *The Internasional Organization of Scientific Research-Journal of Mathematics*, *15*(2), 32-38. https://doi.org/10.9790/5728-1502023238.

Senthilkumar, P., & Rajendran, G. (2011). New approach to solve symmetric fully fuzzy linear systems. *Sadhana*, *36*(6), 933-940. https://doi.org/10.1007/s12046-011-0059-8.

Siahlooei, E., & Fazeli, S.A.S. (2018). An application of interval arithmetic for solving fully fuzzy linear systems with trapezoidal fuzzy numbers. *Advances in Fuzzy Systems*, *2018*. Article ID 2104343. https://doi.org/10.1155/2018/2104343.

Sulaiman, I.M., Mamat, M., Zamri, N., & Ghazali, P.L. (2018). Solving dual fuzzy nonlinear equations via Shamanskii method. *International Journal of Engineering and Technology(UAE)*, *7*(3.28), 89-91. https://doi.org/10.14419/ijet.v7i3.28.20974.

Zeyu, L., Hamdi, A., Belgacem, B., Hana, T., Gholamreza, H., & Ghoushchi, S.J. (2021). An integrated mathematical attitude utilizing fully fuzzy BWM and fuzzy WASPAS for risk evaluation in a SOFC. *Mathematics*, *9*(18), 2328. https://doi.org/10.3390/math9182328.