Optimising Sensor Controlled Systems With Minimal Intervention: A Fuzzy Relational Calculus Approach

Zlatko Vassilev Zahariev *

Posted Date: 4 April 2024

doi: 10.20944/preprints202404.0372.v1

Keywords: Sensor-Controlled Systems; Fuzzy Linear Systems of Equations; Inverse Problem

 doi:10.20944/preprints202404.0372.v1

Article

Optimizing Sensor Controlled Systems With Minimal Intervention: A Fuzzy Relational Calculus Approach

Zlatko Zahariev

Technical University of Sofia, 1756 Sofia, Bulgaria; zlatko@tu-sofia.bg

Abstract: This article describes an approach for optimizing sensor-controlled systems through minimal intervention, utilizing Fuzzy Linear Systems of Equations (FLSE). Staring with a generalized model of the system behavior, incorporating an array of control units, environmental sensors, and an expert knowledge base. The described problems of detecting the level of intervention needed to change the system state to another is handled with the help of developed methods for solving the inverse problem for FLSE. By achieving minimal intervention, we ensure that system adjustments effective, economically optimal and non-intrusive. A MATLAB-based implementation is presented.

Keywords: sensor-controlled systems; fuzzy linear systems of equations; inverse problem

1. Introduction

This article focuses on a type of generalized sensor-controlled systems and introduces a method to alter their behavior while minimizing the level of intervention.

The described systems have the following components:

- An array of manageable control units
- A set of sensors providing information about the environment
- An expert knowledge module defining how the control units can change the environment

In that setup, assuming that we have a well defined expert knowledge, we can easily obtain information about the current status of the environment based on the sensor readings.

Let's assume that now we want to change the environment status. The task, object to this article is to find the minimal possible change to the control units in order to achieve the new target state of the environment. I.e. if a system has a control units set to a state C_1 and the environment is in status E_1, and if we want to achieve environment state E_2, in this article we will answer the question of how to change the control units state to C_2, so that the difference between C_1 and C_2 will be minimal.

Remark 1. *Assuming that we use sensors to measure the environment status, we will use the terms sensor readings, environment status, and system behavior as interchangeable.*

In the following sections we will use a fuzzy rule-based expert knowledge base to represent our knowledge about how the control units influence on the environment. We will solve the inverse problem in fuzzy relational calculus [1–3] to find all possible changes to the control units, leading to the desired environment state, and will select the one closest to the current control units state.

We will use a fuzzy linear systems of equation (FLSE) of type $A \star X = B$ (see Chapter 3).

The fairly direct approach for this task is to use a fuzzy inference system [4–9] and [10] to define our knowledge base as a set of fuzzy if-then rules. Then we can represent those rules as a FLSE (Section 3). Such representation is discussed and implemented in [11]. Another, more candid approach it to directly define the relation between the control units values and the expected sensor reading with FLSE, where the fuzzy matrix (FM) A will represent our knowledge base, the fuzzy matrix X will represent level of activation for the control units, and the matrix B will represent the achieved environment status.

 doi:10.20944/preprints202404.0372.v1

2. Sensor Controlled System Examples

For the sake of completeness, let's look at a few examples of sensor-controlled systems based on the architecture represented by Figure 1. Many other examples can fall in that architecture.

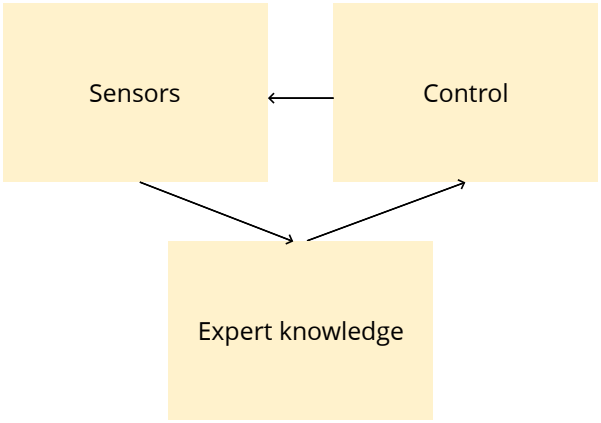

Figure 1. System overview.

2.1. Smart Home System

These systems may use an array of **sensors** to monitor environmental conditions like temperature, humidity, and light, along with manageable **control** units like thermostats, blinds, and lights. The **expert knowledge** module might consist of rules about the homeowner's preferences to adjust settings for optimal comfort and energy efficiency.

2.2. Automated Industrial Manufacturing Line

In these systems, **sensors** may collect data on machine performance (e.g. time to produce a single unit), product quality (e.g. analyzing final product photos), and environmental conditions. **Control** units may include conveyor belts, robotic arms or CNC machines speed control, humidity control, etc. The **expert knowledge module** would use rules to optimize production processes, reduce waste, and maintain quality standards. In separate moments we may need faster production sacrificing on the quality, or better quality sacrificing on the production speed performance.

2.3. Precision Agriculture Systems

These utilize **sensors** to monitor soil moisture, nutrient levels, and crop health. Manageable **control** units might involve irrigation systems, fertilizer spreaders, and spraying drones. The system's **expert knowledge** may include rules to optimize watering schedules, fertilizer application, and pest control, based on the current environment status, ensuring maximal crop yield.

2.4. Health Monitoring Systems

Having **sensors** tracking health status metrics like heart rate, blood oxygen levels, and blood pressure levels. The **control** units could include notification systems or interfaces that communicate with other devices like medication dispensers. The **expert knowledge** implements rules to send alerts or adjust medication takings based on recommendations for the user.

3. Fuzzy Linear Systems of Equation (FLSE)

To represent the sensor-controlled system we will use a FLSE having one of the forms:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 April 2024 doi:10.20944/preprints202404.0372.v1

$$\begin{vmatrix} (a_{11} \rightarrow_{t_r} x_1) \vee (a_{12} \rightarrow_{t_r} x_2) \vee \ldots \wedge (a_{1n} \rightarrow_{t_r} x_n) = b_1 \\ (a_{21} \rightarrow_{t_r} x_1) \vee (a_{22} \rightarrow_{t_r} x_2) \vee \ldots \wedge (a_{2n} \rightarrow_{t_r} x_n) = b_2 \\ \ldots \\ (a_{m1} \rightarrow_{t_r} x_1) \vee (a_{m2} \rightarrow_{t_r} x_2) \vee \ldots \wedge (a_{mn} \rightarrow_{t_r} x_n) = b_m \end{vmatrix} \quad (1)$$

or

$$\begin{vmatrix} (a_{11} \rightarrow_{s_r} x_1) \wedge (a_{12} \rightarrow_{s_r} x_2) \wedge \ldots \wedge (a_{1n} \rightarrow_{s_r} x_n) = b_1 \\ (a_{21} \rightarrow_{s_r} x_1) \wedge (a_{22} \rightarrow_{s_r} x_2) \wedge \ldots \wedge (a_{2n} \rightarrow_{s_r} x_n) = b_2 \\ \ldots \\ (a_{m1} \rightarrow_{s_r} x_1) \wedge (a_{m2} \rightarrow_{s_r} x_2) \wedge \ldots \wedge (a_{mn} \rightarrow_{s_r} x_n) = b_m \end{vmatrix} \quad (2)$$

Where the operations $\wedge$ and $\vee$, are respectively taking minimum and taking maximum, and the operations $\rightarrow_{s_r}$ and $\rightarrow_{t_r}$, are according to the following Table 1 for $r = 1, ..., 3$:

Table 1. $t-$norms and $s-$norms

t-norm	name	expression	s-norm	name	expression
t_3	minimum, Gödel $t-$norm	$t_3(x,y) = \min\{x, y\}$	s_3	maximum, Gödel $t-$conorm	$s_3(x,y) = \max\{x, y\}$
t_2	Algebraic product	$t_2(x,y) = xy$	s_2	Probabilistic sum	$s_2(x,y) = x + y - xy$
t_1	Łukasiewicz $t-$norm	$t_1(x,y) = \max\{x + y - 1, 0\}$	s_1	Bounded sum	$s_1(x,y) = \min\{x + y, 1\}$

A generalized matrix representation of equations (2) and (1) is:

$$A *_{BL} X = B \quad (3)$$

where the fuzzy matrix (FM) A represents our knowledge of the system, the FM X represent the control units settings, and the FM B represents the system behavior. Operation $*_{BL}$ can be any fuzzy multiplication (or fuzzy relational compositions) between A and X. For instance $*_{BL}$ can be $max - min$ (or $s_3 - t_3$), $min - max$ (or $t_3 - s_3$), $max - product$ (or $s_3 - t_2$), etc.

FLSE based on different fuzzy relational compositions are studied in details over the years - see [1,3,12–19]. In what follows we will focus on the $max - min$ composition, which is the most popular one. The presented method will be applicable by analogy for every other fuzzy relation composition in Table 1.

Therefore, we will use FLSE in the form:

$$\begin{vmatrix} (a_{11} \wedge x_1) \vee (a_{12} \wedge x_2) \vee \ldots \vee (a_{1n} \wedge x_n) = b_1 \\ (a_{21} \wedge x_1) \vee (a_{22} \wedge x_2) \vee \ldots \vee (a_{2n} \wedge x_n) = b_2 \\ \ldots \\ (a_{m1} \wedge x_1) \vee (a_{m2} \wedge x_2) \vee \ldots \vee (a_{mn} \wedge x_n) = b_m \end{vmatrix} \quad (4)$$

or

$$A \bullet X = B \quad (5)$$

4. Direct and inverse problems

The set forth in this chapter is according to [15]. The finite FMs $A = (\mu_{ij}^A)_{m \times p}$ and $B = (\mu_{ij}^B)_{p \times n}$ are called ***conformable*** in this order, if the number of columns in A is equal to the number of rows in B.

 doi:10.20944/preprints202404.0372.v1

Definition 1. *Let* $A = (\mu_{ij}^A)_{m\times p}$ *and* $B = (\mu_{ij}^B)_{p\times n}$ *be finite conformable FMs. Obtaining the matrix* $C = (\mu_{ij}^C)_{m\times n}$, *written* $C = A \bullet B$ *is called* $max - min$ ***product*** *of* A *and* B, *if for each* $i,\ j, 1 \le i \le m,\ \ 1 \le j \le n$ *it holds:*

$$\mu_{ij}^C = \max_{k=1}^{p}\Big(min(\mu_{ik}^A, \mu_{kj}^B)\Big), \tag{6}$$

Definition 2. *When the finite conformable FMs* A *and* X *are given, computing the product* $B = A \bullet X$ *is called* ***direct problem resolution*** *for the* $max - min$ *composition of the matrices* A *and* X.

Direct problem is solvable in polynomial time. Software for direct problem resolution is given in [20–22]. It provides all the operations and algorithms described in Table 1 in MATLAB environment.

Definition 3. *When the finite conformable FMs* A *and* B *are given, computing the unknown matrix* X *is called* ***inverse problem resolution*** *for the* $max - min$ *FLSE.*

Inverse problem is solvable in exponential time. Software for inverse problem resolution is also given in [20–22]. It provides solvers for all the compositions in Table 1 in MATLAB environment.

4.1. Solutions of the inverse problem for (5)

For the fuzzy vectors $X = (x_j)_{1\times n}$ and $Y = (y_j)_{1\times n}$ the inequality $X \ge Y$ holds iff $x_j \ge y_j$ for each $j = 1, ..., n$.

A vector $X^0 = (x_j^0)_{1\times n}$ with $x_j^0 \in [0,1]$, $j = 1, ..., n$, is called **solution** of the system (5) if $A \bullet X^0 = B$ holds. The set of all solutions of the system is called **complete solution set** and it is denoted by $\mathbb{X}^0$. If $\mathbb{X}^0 \neq \emptyset$ then the system is called **solvable** (or **consistent**), otherwise it is called **unsolvable** (or **inconsistent**).

A solution $X_u^0 \in \mathbb{X}^0$ is called *upper solution* if for any $X^0 \in \mathbb{X}^0$ the inequality $X_u^0 \le X^0$ implies $X^0 = X_u^0$. A solution $X_{low}^0 \in \mathbb{X}^0$ is called *lower solution* if for any $X^0 \in \mathbb{X}^0$ the inequality $X^0 \le X_{low}^0$ implies $X^0 = X_{low}^0$. If the upper solution is unique, it is called *greatest* (or *maximum*) solution. The n-tuple $(X_1, ..., X_n)$ with $X_j \subseteq [0,1]$ is called *interval solution* if any $X^0 = (x_j^0)_{n\times 1}$ with $x_j^0 \in X_j$ for each $j = 1, ..., n$ implies $X^0 = (x_j^0)_{n\times 1} \in \mathbb{X}^0$. Any interval solution whose components (interval bounds) are determined by the greatest solution from the right and by a lower solution from the left, is called *maximal interval solution* of (5).

It is well known [1,15], that any solvable $max - min$ fuzzy linear system of equations has unique greatest solution and one or many lower solutions. All the solutions in between the greatest solution and any of the lower solutions are also a solutions of (5). In order to find all solutions of the solvable system (5), it is necessary to find both its greatest solution and all of its lower solutions. Finding the greatest solution is relatively simple task often used as a criteria for establishing solvability of the system [14]. Finding all lower solutions is much more complex (from computational point of view) task, and is reasonable only when the lowest solution exists.

Efficient algorithms for finding the complete solution set of (5) are given in [20–22]. A software implementations are given in [3,21,22]. We will use the software implementation from [22].

5. Adjusting the Sensor Control System Behavior with Minimal Intervention

Let's have a sensor controlled system called S, described by fuzzy linear system of equation (5): $A \bullet X = B$. We can obtain its current behavior (or environment status) through the control units status and the knowledge base. When we represent our knowledge base with the fuzzy matrix A, and our control units settings with the fuzzy vector X - we can either solve the direct problem in FLSE to obtain the current system status or simply check it's sensor readings. In case we want to adjust (or change) its behavior we will need to solve the inverse problem. I.e. we can describe the new, target behavior through the fuzzy vector B. Then we can solve the inverse problem in FLSE in order to achieve one or

 doi:10.20944/preprints202404.0372.v1

many possible ways to achieve this new status, then we need to adjust the system, so that the new control units settings cover one of the possible solutions of the inverse problem. Finally, if we want to achieve minimal adjustment to the system and still achieve the new target status, we need to choose such a solution of the inverse problem, so that the difference between the current control status and the target status is minimal. This process is described in the following algorithm:

Algorithm 1 Change system status with minimal intervention

Step 1: Input the initial parameters for the fuzzy matrix A, and the fuzzy vector $X = X_0$
Step 2: (Optional) Predict the current system status - Solve $A \bullet X_0 = B_0$ where A and X_0 are known. In that case B_0 will describe the current system behavior.
Step 3: Define the new target status of the system as a fuzzy vector B_1
Step 4: Solve the inverse problem for the system $A \bullet X = B_0$, where we have the FM A, and the new target status B
Step 5: Obtain all possible solutions for the fuzzy vector $X = X_{sol}$
Step 6: **IF** the system is not consistent - the target status cannot be obtained. Go to Step 9, **ELSE** go to Step 7
Step 7: Find a possible value form $X_{sol} = X_{sol_{min}}$, such that $|X_0 - X_{sol_{min}}|$ is minimal among all the possible values from X_{sol}
Step 8: Using any kind of actuators, change the system in such way, so the new sensor readings X are equal to $X_{sol_{min}}$
Step 9: END

Remark 2. *Step 2 is given for completeness. In most cases more accurate information can be obtained directly from the sensor readings. However, it can be a valid step if we experience sensor malfunction, or if we want to validate either of out knowledge base or the accuracy of the sensors.*

Algorithm 1, while straightforward, includes several steps that require further elaboration.

5.1. Step 1: Detect Current System Status

This step is described by Definition 1 and Definition 2. To approach this task, we will simply need to calculate the $max - min$ product of the matrices A and X_0. In the following example, we will utilize the software from [22] for this calculations.

5.2. Step 3: Solve the Inverse Problem

This step is the most complex for the Algorithm 1. It is subject of great scientific interest. The main results are published in [1–3]. They demonstrate long and difficult period of investigations for discovering analytical methods and procedures to determine the complete solution set, as well as to develop software for computing solutions. The first and most essential are Sanchez results [23] for the greatest solution of fuzzy relational equations with max-min and min-max composition. Sanchez gave formulas that permit to determine the potential greatest solution in any of these cases, often used as solvability criteria. Universal algorithm and software for solving max-min and min-max fuzzy relational equations is proposed in [3], and [13]. The relationship with the covering problem is subject of [12]. Systems of fuzzy relation equations and fuzzy relation inequalities, based on different fuzzy relational compositions are studied in details in [1,3,12–19].

The most advance software implementation for solving the inverse problem is developed by the author [15,21,22]. It includes algorithm implementation for eight popular fuzzy composition (including the $max - min$ composition). In the following examples we will use the software from [22] to find the complete solution set for the inverse problem for the system $A \bullet X = B$. This software features a great performance in finding the complete solution set, even for relatively large systems. For more information about the computational and memory complexity, as well as experimental results and comparison with the other available software, see [22].

 doi:10.20944/preprints202404.0372.v1

5.3. Step 6: Find the Closest to the Current State, Solution

For the system 5, every interval described by the greatest solution from the right the any of its lower solutions from the left is called **interval solution**. Any value inside the interval solution is still a solution of the system [1,15]. Therefore the system 5, has as many interval solutions as many lower solutions. In order to find the solution (as a fuzzy vector), which is closest to some other fuzzy vector (in our case, the initial control units settings) we can use the following straightforward algorithm:

Algorithm 2 Find the closest solution

Step 1: Define the current sensor readings X_0
Step 2: **For** every interval solution $[X_{low_i}, X_{gr}]$:
Step 3: Initialize a candidate solution $\overline{X}_i$
Step 4: **For** every dimension (j) of the i^{th} vector:
Step 5: **If** X_{0_j} is inside the interval $[X_{low_{ij}}, X_{gr_j}]$
Step 6: **Then** X_{0_j} is already in the target interval, so $\overline{X}_{ij} = X_{0_j}$
Step 7: **Else** for $\overline{X}_{ij}$ take either $X_{low_{ij}}$ or X_{gr_j}, which is closer to the target $\overline{X}_{ij}$.
End If
End For
End For
Step 8: From all candidate solutions $\overline{X}_i$, the new X_{new} is the one which is mathematically closest to the current X_0. I.e. select candidate solution $\overline{X}_i$, such that $\sum_j |\overline{X}_{i_j} - X_{0_j}|$ is minimal.
Step 9: **END**

6. Example and MATLAB Execution

Example 1. *Let's have a max − min FLSE. It's knowledge base will be represented by the FM A:*

$$A = \begin{pmatrix} 0.5 & 0.6 & 0.4 & 0.1 & 0.4 \\ 0.1 & 0.4 & 0.1 & 0.2 & 0 \\ 0.9 & 0.5 & 0.2 & 0.2 & 0.9 \end{pmatrix} \tag{7}$$

This system has 5 sensors modeling the environment status a 3 rules describing how the environment parameters (B) will change depending on the values (normalized as a fuzzy number) of the sensors (X). Each sensor reading have it's own level of importance in each rule.

In this example we will use the following sensor readings as a starting point for the system:

$$X = \begin{pmatrix} 0.9 \\ 0.5 \\ 0.5 \\ 0.3 \\ 0.9 \end{pmatrix} \tag{8}$$

Let's implement Algorithm 1. We will use the software from [22] and will execute the algorithm steps in MATLAB.

6.1. Algorithm 1, Step 1 - Initialize FMs

MATLAB Session 1

```
>> A = fuzzyMatrix([.5 .6 .4 .1 .4; .1 .4 .1 .2 0; .9 .5 .2 .2 .9])
A =
  3x5 fuzzyMatrix:
  double data:
    0.5000    0.6000    0.4000    0.1000    0.4000
    0.1000    0.4000    0.1000    0.2000         0
    0.9000    0.5000    0.2000    0.2000    0.9000

>> X = fuzzyMatrix([.9; .5; .5; .3; .9])
X =
  5x1 fuzzyMatrix:
```

 doi:10.20944/preprints202404.0372.v1


```
  double data:
    0.9000
    0.5000
    0.5000
    0.3000
    0.9000
```

6.2. *Algorithm 1, Step 2 - Predict the Environment Status*

MATLAB Session 2

```
>> B = maxmin(A,X)
B =
  3x1 fuzzyMatrix:
  double data:
    0.5000
    0.4000
    0.9000
```

The system has 3 environment parameters. By executing this step, we can see in what degree every ot this parameters is currently observed.

6.3. *Algorithm 1, Step 3 - Define New Target System Status*

Now let's assume that we want to achieve another environment status by lowering the level of degree with witch the third parameter is present ($B_3 = 0.9$). Let's try with two options - B_{new_1}, and B_{new_2} for which b_1 and b_2 will remain the same, but we will try it lower b_3 first to 0.4, and then to 0.5

MATLAB Session 3

```
>> B_new_1 = fuzzyMatrix([.5; .4; .4])
B_new_1 =
  3x1 fuzzyMatrix:
  double data:
    0.5000
    0.4000
    0.4000

>> B_new_2 = fuzzyMatrix([.5; .4; .5])
B_new_2 =
  3x1 fuzzyMatrix:
  double data:
    0.5000
    0.4000
    0.5000
```

6.4. *Algorithm 1, Step 4 - Solve the Inverse Problem'*

Now we create two FLSE (one for B_{new_1} and one for B_{new_2}, and solve them. The accepted parameters for the *solve_inverse* function are:

1. The composition of the FLSE - In our case this will be $max - min$
2. The matrix A
3. The matrix B
4. The matrix X - this is what we will try to find, so we can pass an empty array here
5. A Boolean parameter to flag if we need to find the complete solution set, or just the greatest solution

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 April 2024 doi:10.20944/preprints202404.0372.v1

We solve both of the systems:

MATLAB Session 4

```
>> S1 = fuzzySystem('maxmin', A, B_new_1, [], true)
S1 =
  fuzzySystem with properties:
     composition: 'maxmin'
               a: [3x5 fuzzyMatrix]
               b: [3x1 fuzzyMatrix]
               x: [5x1 fuzzyMatrix]
            full: 1
    inequalities: 0

>> S1.solve_inverse
ans =
  fuzzySystem with properties:
     composition: 'maxmin'
               a: [3x5 fuzzyMatrix]
               b: [3x1 fuzzyMatrix]
               x: [1x1 struct]
            full: 1
    inequalities: 0

>> S2 = fuzzySystem('maxmin', A, B_new_2, [], true)
S2 =
  fuzzySystem with properties:
     composition: 'maxmin'
               a: [3x5 fuzzyMatrix]
               b: [3x1 fuzzyMatrix]
               x: [5x1 fuzzyMatrix]
            full: 1
    inequalities: 0

>> S2.solve_inverse
ans =
  fuzzySystem with properties:
     composition: 'maxmin'
               a: [3x5 fuzzyMatrix]
               b: [3x1 fuzzyMatrix]
               x: [1x1 struct]
            full: 1
    inequalities: 0
```

6.5. Algorithm 1, Step 5 - Obtain Solutions

First lets obtain all the solutions for B_{new_1} (i.e. S_1)

MATLAB Session 5

```
>> S1.x
ans =
  struct with fields:
          rows: 3
          cols: 5
          help: [3x5 fuzzyMatrix]
            gr: [5x1 double]
```

 doi:10.20944/preprints202404.0372.v1


```
          ind: [3x1 double]
        exist: 0
    contradict: 1
```

For S_1 we can see that *S1.x.exists = 0*, which means that the system is inconsistent (i.e. the target environment state cannot be obtained).

MATLAB Session 6

```
>> S2.x
ans =
  struct with fields:
         rows: 3
         cols: 5
         help: [2x5 fuzzyMatrix]
           gr: [5x1 fuzzyMatrix]
          ind: [3x1 double]
        exist: 1
    dominated: 3
    help_rows: 2
          low: [5x2 fuzzyMatrix]
```

For S_2 we can see that the system is consistent *S2.x.exists = 1*, and it has 1 greatest solution (*S2.x.gr.size(2) = 1*), and 2 lower solutions (*S2.x.low.size(2) = 2*).

6.6. Algorithm 1, Step 6 - Check for Consistency

For the system S_1 we already determined that it is inconsistent so the algorithm ends here. Further we will focus on the system S_2.

6.7. Algorithm 1, Step 7 - Find the new control units settings

Let's print the greatest and all the lower solutions of the system S_2

MATLAB Session 7

```
>> S2.x.gr
ans =
  5x1 fuzzyMatrix:
  double data:
    0.5000
    0.5000
    1.0000
    1.0000
    0.5000

>> S2.x.low
ans =
  5x2 fuzzyMatrix:
  double data:
    0.5000         0
    0.4000    0.5000
         0         0
         0         0
         0         0
```

From the greatest solution and the two lower solutions we know that the system has two interval solution, which determines the complete solution set of S_2:

 doi:10.20944/preprints202404.0372.v1

$$X_1 = \begin{pmatrix} 0.5 \\ [0.4, 0.5] \\ [0, 1] \\ [0, 0.1] \\ [0, 0.5] \end{pmatrix} \quad X_2 = \begin{pmatrix} [0, 0.5] \\ 0.5 \\ [0, 1] \\ [0, 1] \\ [0, 0.5] \end{pmatrix} \tag{9}$$

Let's find the closest to the initial X vector for both of the interval solutions:

MATLAB Session 8

```
>> closestVector_1 = min(max(X, S2.x.low(:,1)), S2.x.gr)
closestVector_1 =
    0.5000
    0.5000
    0.5000
    0.3000
    0.5000

>> closestVector_2 = min(max(X, S2.x.low(:,2)), S2.x.gr)
closestVector_2 =
    0.5000
    0.5000
    0.5000
    0.3000
    0.5000
```

We can see that it is the same vector for both interval solutions, so the new value for X should be:

$$X_{new} = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.3 \\ 0.5 \end{pmatrix} \tag{10}$$

That means that we will need to adjust just the first control unit, lowering it with 0.4 and the last control unit setting lowering it again with 0.4. This it the minimal intervention that we can achieve in order to change the system behavior from B, to B_{new_2}.

6.8. Algorithm Step 8

Depending on the real world implementation of the system, we now need to use our actuators connected to the first and the last control units, in order to adjust them to the target values.

7. Conclusion

In this article, a method for adjusting sensor-controlled systems with minimal intervention, is presented. It facilitates the methods for solving the direct and the inverse problems for fuzzy linear systems of equations (FLSE). While here we have a generalized approach, it should be relatively easy to use the same principles for mode specific systems. Using the presented here algorithms, we can ensure that the change we want to achieve is either not possible, or if possible - done with the minimal possible intervention. This is ensured from the algorithms backing the process of acquiring the complete solution set of the system (5), presented in [21].

For the software implementation of the provided here example, we use the software from [22], which is already proven to work efficiently, and able to find all possible lower solutions of the system - a complex, non-trivial task.

 doi:10.20944/preprints202404.0372.v1

The most obvious next step in this research will be to introduce a method, as well as a way to evaluate, a new state of the system, which is closest to the target, in the cases, when the target in not possible (i.e. when the system (5) is not consistent.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

FLSE Fuzzy Linear Systems of Equations
FM Fuzzy Matrix

References

1. B. De Baets, Analytical solution methods for fuzzy relational equations, in the series: Fundamentals of Fuzzy Sets, The Handbooks of Fuzzy Sets Series, D. Dubois and H. Prade (eds.), Vol. 1, Kluwer Academic Publishers, 2000, pp. 291-340.
2. A. Di Nola, W. Pedrycz, S. Sessa and E. Sanchez, Fuzzy Relation Equations and Their Application to Knowledge Engineering, Kluwer Academic Press, Dordrecht, 1989.
3. K. Peeva and Y. Kyosev, Fuzzy Relational Calculus-Theory, Applications and Software (with CD-ROM), In the series Advances in Fuzzy Systems - Applications and Theory, Vol. 22, World Scientific Publishing Company, 2004. Software downloadable from http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=6214
4. J.F. Baldwin, "Fuzzy logic and fuzzy reasoning," in Fuzzy Reasoning and Its Applications, E.H. Mamdani and B.R. Gaines (eds.), London: Academic Press, 1981.
5. F. Esragh and E.H. Mamdani, "A general approach to linguistic approximation," in Fuzzy Reasoning and Its Applications, E.H. Mamdani and B.R. Gaines (eds.), London: Academic Press, 1981.
6. J.-S. R. Jang and N. Gulley, The Fuzzy Logic Toolbox for Use with MATLAB, Massachusetts: MathWorks Inc., 1995.
7. Mamdani EH, Assilian S (1975) An experiment in linguist synthesis with fuzzy logic controller. Int J Man-Mach Stud 7(1):1-13.
8. Siler W, Buckley JJ (2004) Fuzzy expert systems and fuzzy reasoning. Wiley InterScience.
9. Yager, R.R., and Zadeh, L. A., "An Introduction to Fuzzy Logic Applications in Intelligent Systems", Kluwer Academic Publishers, 1991.
10. L.A. Zadeh, "Making computers think like people," I.E.E.E. Spectrum, 8/1984, pp. 26-32.
11. Z. Zahariev, Fuzzy reasoning through fuzzy linear systems of equations, “Proceedings of the Technical University - Sofia”, Volume 60, book 3, 58-66, 2010, ISSN 1311-0829.
12. A. Markovskii, On the relation between equations with max-product composition and the covering problem, Fuzzy Sets Systems, 153, 2005, pp. 261–273
13. K. Peeva, Universal algorithm for solving fuzzy relational equations, Italian Journal of Pure and Applied Mathematics 19, 2006, pp. 9-20.
14. K. Peeva, Resolution of Fuzzy Relational Equations – Method, Algorithm and Software with Applications, Information Sciences, 234 (Special Issue), 2013, 44-63, doi 10.1016/j.ins.2011.04.011, ISSN 0020-0255.
15. K. Peeva, G. Zaharieva, and Z. Zahariev, Resolution of max-t-norm fuzzy linear system of equations in BL-algebras, AIP Conference Proceedings vol. 1789, 060005, 2016, doi: 10.1063/1.4968497
16. J. Ignjatović, M. Ćirić, Br. Šešelja, A. Tepavčević, Fuzzy relational inequalities and equations, fuzzy quasi-orders, closures and openings of fuzzy sets, Fuzzy Sets and Systems Volume 260, 2015, 1-24
17. S. Yang, Some Results of the Fuzzy Relation Inequalities With Addition–Min Composition, IEEE Transactions on Fuzzy Systems, vol. 26, no. 1, 239-245, 2018, doi: 10.1109/TFUZZ.2017.2648864.
18. K. Peeva, Universal algorithm for solving fuzzy relational equations. Journal De Mathematiques Pures Et Appliquees 19, 2006, 169–188
19. X. Yang, Solutions and strong solutions of min-product fuzzy relation inequalities with application in supply chain, Fuzzy Sets and Systems, 384, 2020, 54–74

 doi:10.20944/preprints202404.0372.v1

20. Z. Zahariev, Solving Max-min Relational Equations. Software and Applications, in International Conference "Applications of Mathematics in Engineering and Economics (AMEE'08)", AIP Conference Proceedings, vol. 1067, G. Venkov, R. Kovatcheva, V. Pasheva (eds.), American Institute of Physics, 2008, 516-523.
21. Z. Zahariev, Software package and API in MATLAB for working with fuzzy algebras, in International Conference "Applications of Mathematics in Engineering and Economics (AMEE'09)", AIP Conference Proceedings, vol. 1184, G. Venkov, R. Kovatcheva, V. Pasheva (eds.), American Institute of Physics, ISBN 978-0-7354-0750-9, 2009, 434-350.
22. Z. Zahariev, http://www.mathworks.com/matlabcentral/fileexchange/27046-fuzzy-calculus-core-fc2ore, 2024 (most recent update)
23. E. Sanchez, Resolution of composite fuzzy relation equations, Information and Control, 30, 1976, pp. 38-48.