**Introduction to MathWorks: Fuzzy Logic Toolbox & Simulink Demo**

**What is Fuzzy Logic?**

Fuzzy Logic is a form of logic that deals with reasoning that is approximate rather than fixed and exact. It mimics human decision-making by working with **degrees of truth** rather than binary true/false values.

For example:

* Instead of saying: *“Temperature > 30°C is hot”*,  
  Fuzzy logic says: *“Temperature is somewhat hot”* or *“very hot”* based on membership levels.

**What is MathWorks?**

**MathWorks** is a company that develops:

* **MATLAB** (MATrix LABoratory): A high-level programming environment for numerical computation, visualization, and algorithm development.
* **Simulink**: A block-diagram environment for simulation and model-based design.

**What is the Fuzzy Logic Toolbox?**

The **Fuzzy Logic Toolbox** in MATLAB provides tools for designing systems based on fuzzy logic rules. It helps you:

* Build fuzzy inference systems (FIS)
* Define input/output membership functions
* Set up fuzzy rules
* Simulate and analyze fuzzy logic behavior
* Integrate fuzzy logic with control systems

**Key Features:**

| **Feature** | **Description** |
| --- | --- |
| **FIS Editor** | Create a new fuzzy inference system (Mamdani or Sugeno) |
| **Membership Function Editor** | Define fuzzy sets with triangular, trapezoidal, Gaussian shapes, etc. |
| **Rule Editor** | Build logic rules like “If A is Low and B is High, then Output is Medium” |
| **Rule Viewer & Surface Viewer** | Visualize fuzzy rule results and inference surfaces |
| **Integration with Simulink** | Add fuzzy logic blocks to dynamic system models |

**Fuzzy Logic Simulink Demo Overview**

You can model fuzzy logic systems in **Simulink** using the **Fuzzy Logic Controller** block.

**Basic Simulation Example:**

**System: Fan Control Based on Temperature**

1. **Inputs**: Temperature (Low, Medium, High)
2. **Output**: Fan Speed (Slow, Medium, Fast)
3. **Rules**:
   * If Temperature is Low → Fan Speed is Slow
   * If Temperature is High → Fan Speed is Fast

**Simulink Block Setup:**

| **Block** | **Purpose** |
| --- | --- |
| Constant Block | Simulates input temperature |
| Fuzzy Logic Controller | Processes fuzzy rules |
| Scope Block | Displays fan speed output |
| Gain, Sum, Mux | Signal processing (optional) |

**Running the Demo in MATLAB (Code-based):**

fuzzyLogic = readfis('fanControl.fis'); % Load FIS

tempInput = 35; % Example input

fanSpeed = evalfis(fuzzyLogic, tempInput); % Output

disp(fanSpeed);

**Applications of Fuzzy Logic Toolbox**

* Control systems (AC/fan speed, vehicle braking)
* Industrial automation
* Decision-making systems
* Robotics and AI systems

**Demo Video Suggestion**

You can show a sample from:  
<https://www.mathworks.com/products/fuzzy-logic.html>

# **Fuzzy Logic Practical using MATLAB and Simulink – Step-by-Step**

## ****Objective of the Practical****

Design and simulate a **Fuzzy Logic Controller** to control the speed of a **fan** based on **room temperature**, using:

* MATLAB’s **Fuzzy Logic Toolbox**
* **Simulink** environment

## ****Pre-requirements****

* MATLAB with **Fuzzy Logic Toolbox** and **Simulink**
* OR access to **MATLAB Online** (with licensed login)
* Basic understanding of input/output systems and rule-based logic

## ****Part 1: Create a Fuzzy Inference System (FIS)****

### Step 1: Open Fuzzy Logic Designer

* In MATLAB command window, type:
* fuzzy

→ This opens the **FIS Editor (GUI)**

### Step 2: Add Input Variable — Temperature

1. Click on **"Edit" → "Add Variable → Input"**
2. Rename it to **"Temperature"**
3. Click on **"View → Membership Functions"**
4. Define 3 membership functions:
   * **Low** (e.g., 0–15°C)
   * **Medium** (e.g., 10–30°C)
   * **High** (e.g., 25–45°C)
5. Use **Triangular (trimf)** or **Trapezoidal (trapmf)** shapes

### Step 3: Add Output Variable — Fan Speed

1. Click **"Edit" → "Add Variable → Output"**
2. Rename it to **"FanSpeed"**
3. Set range (e.g., 0–100 RPM or %)
4. Add membership functions:
   * **Slow**
   * **Medium**
   * **Fast**

### Step 4: Define Fuzzy Rules

Go to **Rule Editor** and enter rules like:

1. IF Temperature is **Low** THEN FanSpeed is **Slow**
2. IF Temperature is **Medium** THEN FanSpeed is **Medium**
3. IF Temperature is **High** THEN FanSpeed is **Fast**

Click **“Add Rule”** and then **“Evaluate Rules”**.

### Step 5: Save FIS

* Click **“File → Save As”**
* Save as fanController.fis

## ****Part 2: Simulate Fuzzy Logic Controller in Simulink****

### Step 6: Open Simulink

* In MATLAB, type:
* simulink
* Create a **New Model**

### Step 7: Add Required Blocks

Drag the following blocks from Simulink Library:

| **Block** | **Library Location** |
| --- | --- |
| **Constant** | Sources |
| **Fuzzy Logic Controller** | Fuzzy Logic Toolbox |
| **Scope** | Sinks |

### Step 8: Configure Blocks

### Constant Block

* Set value (e.g., 30 for Temperature)

#### Fuzzy Logic Controller

* Double-click the block
* Click **Browse** to load your fanController.fis

#### Scope

* Used to view the output (FanSpeed)

### Step 9: Connect the Blocks

Make the following connections:

Constant → Fuzzy Logic Controller → Scope

### Step 10: Run the Simulation

* Click **"Run"** (green play button)
* Open the **Scope block** to see the output (Fan Speed)

## ****Optional: Use Rule Viewer****

If using MATLAB GUI:

* Type fuzzy(fis)
* Use **Rule Viewer** and **Surface Viewer** to test different inputs

## ****Expected Output****

* Input: Temperature (e.g., 30°C)
* Output: Fan Speed (e.g., 70%)

The output is **not binary**; it reflects **degrees** based on fuzzy rules.

## ****Conclusion****

This practical shows:

* How fuzzy systems handle imprecise input (like "Hot")
* Real-world applications of FIS in control systems
* Integration of rule-based logic in dynamic models using Simulink

Sure Meghaa! Let's go through **two complete examples** using MATLAB's **Fuzzy Logic Toolbox**, each with detailed **explanations**.

**✅ Example 1: Washing Machine Timer Controller**

**🎯 Objective:**

Create a fuzzy logic system that determines **Wash Time** based on:

* **Dirtiness** of clothes (0 to 10)
* **Load Size** (0 to 10)

**🧩 Step-by-Step Explanation:**

**Step 1: Define the Problem**

We want the washing machine to increase wash time if:

* Clothes are dirtier
* Load size is bigger

So, the output (Wash Time) depends on:

* Dirtiness (input 1)
* Load Size (input 2)

**Step 2: Open FIS Editor**

In MATLAB command window:

fuzzy

It opens the **Fuzzy Inference System (FIS) Editor**.

**Step 3: Define Inputs and Output**

* Input 1: Dirtiness (range: 0 to 10)
* Input 2: LoadSize (range: 0 to 10)
* Output: WashTime (range: 0 to 60 minutes)

**Step 4: Define Membership Functions**

**Input: Dirtiness**

* Low → triangle (0, 0, 5)
* Medium → triangle (3, 5, 7)
* High → triangle (5, 10, 10)

**Input: Load Size**

* Small → triangle (0, 0, 5)
* Medium → triangle (3, 5, 7)
* Large → triangle (5, 10, 10)

**Output: Wash Time**

* Short → triangle (0, 0, 20)
* Medium → triangle (15, 30, 45)
* Long → triangle (40, 60, 60)

**Step 5: Create Fuzzy Rules**

Use **Rule Editor** in the FIS window:

| **Rule No** | **IF Dirtiness** | **AND LoadSize** | **THEN WashTime** |
| --- | --- | --- | --- |
| 1 | Low | Small | Short |
| 2 | Medium | Medium | Medium |
| 3 | High | Large | Long |
| 4 | Low | Large | Medium |
| 5 | High | Small | Medium |

You can add these using dropdowns in the Rule Editor GUI.

**Step 6: Simulate Using Rule Viewer**

In **Rule Viewer**:

* Set Dirtiness = 8
* Set LoadSize = 6
* It might show WashTime ≈ 45 minutes

Use **Surface Viewer** to see the 3D relationship between all inputs and output.

**Step 7: Run in MATLAB Code**

Once saved as wash.fis, test it with code:

fis = readfis('wash');

output = evalfis([8 6], fis); % Dirtiness = 8, LoadSize = 6

disp(['Recommended Wash Time: ', num2str(output), ' minutes']);

**✅ Example 2: Student Performance Evaluation**

**🎯 Objective:**

Evaluate a student’s performance based on:

* **Attendance** (in %)
* **Assignment Marks** (out of 10)

**🧩 Step-by-Step Explanation:**

**Step 1: Define the Problem**

We want to suggest the student’s **performance level** based on:

* Attendance (0% to 100%)
* Assignment score (0 to 10)

**Step 2: Launch FIS Editor**

fuzzy

**Step 3: Define Inputs and Output**

* Input 1: Attendance (0–100)
* Input 2: AssignmentMarks (0–10)
* Output: Performance (0–10)

**Step 4: Set Membership Functions**

**Attendance:**

* Low → triangle (0, 0, 50)
* Average → triangle (40, 60, 80)
* High → triangle (70, 100, 100)

**AssignmentMarks:**

* Poor → triangle (0, 0, 5)
* Good → triangle (3, 5, 7)
* Excellent → triangle (6, 10, 10)

**Performance:**

* Weak → triangle (0, 0, 5)
* Average → triangle (3, 5, 7)
* Strong → triangle (6, 10, 10)

**Step 5: Define Rules**

| **Rule No** | **IF Attendance** | **AND Assignment** | **THEN Performance** |
| --- | --- | --- | --- |
| 1 | Low | Poor | Weak |
| 2 | High | Excellent | Strong |
| 3 | Average | Good | Average |
| 4 | High | Poor | Average |
| 5 | Low | Excellent | Average |

**Step 6: Test in Rule Viewer**

Try:

* Attendance: 90%
* Assignment: 8  
  It may show performance ≈ 9 (Strong)

**Step 7: Use in Code**

Save as student\_perf.fis and run:

fis = readfis('student\_perf');

output = evalfis([90 8], fis); % Attendance = 90%, Assignment = 8

disp(['Performance Rating: ', num2str(output)]);

**✅ Summary of Key Tools in MATLAB Fuzzy Toolbox:**

| **Tool** | **Purpose** |
| --- | --- |
| FIS Editor | Structure system (inputs/outputs) |
| MF Editor | Define shapes and ranges of inputs/outputs |
| Rule Editor | Add fuzzy rules |
| Rule Viewer | Simulate inputs and see output flow |
| Surface Viewer | 3D visualization of logic surface |

Would you like a **.fis file** of these examples or a **PDF handout with screenshots**?

I can also guide you in:

* Creating a fuzzy controller in **Simulink**
* Creating fuzzy systems using **MATLAB code only** (without GUI)

Let me know your preference.