International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-13 Issue-3, September 2024

A Study on Effective Teaching of Fuzzy Controller Design Using ChatGPT

Dong Hwa Kim

***Abstract**: This paper deals with an effective teaching method of fuzzy controller design using ChatGPT and Python in the University. Currently, ChatGPT and related technology penetrate into all areas such as text, image, video, as well as AI areas and industry. ChatGPT technologies are strongly connected with the education of universities and K-12 because of easy using and correct response. There are two categories of AI-related education. The first one is AI-supported education; another thing is education (teaching and learning) to teach AI. In any case, AI and ChatGPT method should be taught with theory and using method. It should be performed with S/W for student's understanding. This paper provides a method on how ChatGPT can be well used fuzzy controller design in education. To illustrate ChatGPT based fuzzy controller design, this paper uses ChatGPT prompt and suggests a good prompt skill.*

***Keywords**: ChatGPT, Prompt, Fuzzy logic, Pendulum, AI, Education*

I. INTRODUCTION

ChatGPT-related technologies are coming everywhere because of easy and correct applications. For teaching and learning at university, teachers have to prepare this ChatGPT prompt and application contents [2, 3, 13, 14, 15].

After the released ChatGPT by OpenAI in late Nov. 22, the ChatGPT application is widely published in articles for essay writing and image production. When they open ChatGPT, several organizers decline its technologies because they worry about ethical issues [11].

At the first time, many school blocked the ChatGPT website because of too much worrying about student study knowledge. For example, LAUSD (https://www.lausd.org/domain/4), one of the largest school districts in the US), immediately blocked access to OpenAI's website. Others such as, Washington, New York, Alabama, and Virginia in the United States and New South Wales in Australia joined.

ChatGPT provides many materials for self-study to the public by the easy question of several words and we can obtain useful material for our research and study or making a good life depending on the prompt [14].

Also, several educational-tech companies such as, Duolingo and Quizlet, Riiid, Teachmint, Carnegie Learning, Udacity, Memrise, Amira Learning, Squirrel AI, Unacademy, and DreamBox Learning start for providing study helping by ChatGPT as start-up companies. Currently, half of all high school students in the US have already integrated OpenAI's chatbot into practice assessments and references [15]. And OpenAI has worked with educators to put together a fact sheet about ChatGPT's potential impact in schools. Many companies are also consulting educators.

Several leading universities in the UK, including Imperial College London and the University of Cambridge, issued statements that warned students against using ChatGPT to cheat. However, the megatrend is on-going [5, 6, 7, 8, 12].

This paper's main aim is to provide teaching and learning material on fuzzy control system design and analysis method, and effective application method of ChatGPT [1].

II. STUDY BACKGROUND

Currently, a new paradigm of education is coming as a wider shift in the role of teachers play in the classroom because of online and chatbots. What educators must now do is show students not only how to find it, but what information to trust and what not to, and how to tell the difference. Teachers are no longer the main role of information and teaching skills. An important thing of many roles of a teacher is to find which one is correct information and effective study method for the fact [4-9].

However, currently, many things of ChatGPT have an impact on many areas, and confusing about how it is a good thing and which one is correct information. Even though many start-ups and many in educators as well as engineers interested in ChatGPT-based work and study. That is, the education system will be doing under this related technology and AI [9, 10, 16, 17].

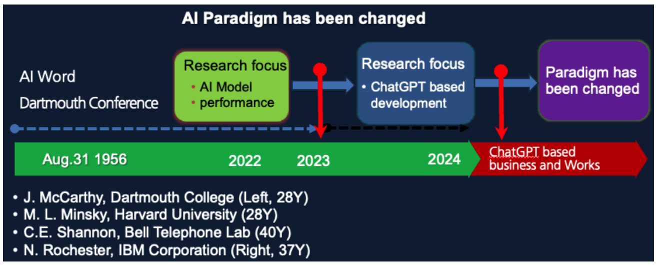

Figure. 1: The Change of AI Research Paradigm After ChatGPT Release

Many teachers now recognize that they have an obligation to learn their easier works using by easier too like ChatGPT [10, 13].

Manuscript received on 19 July 2024 | Revised Manuscript received on 31 July 2024 | Manuscript Accepted on 15 September 2024 | Manuscript published on 30 September 2024.
*Correspondence Author(s)
Dong Hwa Kim*, DSTSC (Daeduck S&T Social Cooperation, S. Korea. E-mail: koreahucare@gmail.com, ORCID ID: 0000-0002-0528-6736

ChatGPT is also one of the social and technology mega trends. It will combine AI and culture and produce a very good idea for the future paradigm of the job. Of course, students should understand well the relationship between many areas and ChatGPT. By using AI and ChatGPT, students should understand the application method of AI more than any other way.

III. IMPLEMENTATION OF CHATGPT-BASED PENDULUM FUZZY CONTROLLER DESIGN

We can analyze the fuzzy control through ChatGPT and Code. We must show on how ChatGPT can be used in technology.

A. ChatGPT Experiment

This paper tried to experiment the fuzzy controller design of pendulum by using ChatGPT. The main target is to find an adaptable prompt to design.

```python
import numpy as np
import matplotlib.pyplot as plt

# Triangular Membership Function
def triangular_mf(x, a, b, c):
    return np.maximum(np.minimum((x-a)/(b-a), (c-x)/(c-b)), 0)

# Trapezoidal Membership Function
def trapezoidal_mf(x, a, b, c, d):
    return np.maximum(np.minimum(np.minimum((x-a)/(b-a), 1), (d-x)/(d-c)), 0)

# Gaussian Membership Function
def gaussian_mf(x, mean, sigma):
    return np.exp(-0.5 * ((x - mean) / sigma) ** 2)

# Bell-Shaped Membership Function
def bell_mf(x, a, b, c):
    return 1 / (1 + np.abs((x - c) / a) ** (2 * b))

# Create an array of x values
x = np.linspace(0, 10, 1000)

# Parameters for the membership functions
tri_params = (2, 5, 8)
trap_params = (1, 3, 7, 9)
gauss_params = (5, 1.5)
bell_params = (2, 4, 5)
```

```python
# Calculate the membership values
tri_mf = triangular_mf(x, *tri_params)
trap_mf = trapezoidal_mf(x, *trap_params)
gauss_mf = gaussian_mf(x, *gauss_params)
bell_mf = bell_mf(x, *bell_params)

# Plotting the membership functions
plt.figure(figsize=(10, 6))

plt.plot(x, tri_mf, label='Triangular MF')
plt.plot(x, trap_mf, label='Trapezoidal MF')
plt.plot(x, gauss_mf, label='Gaussian MF')
plt.plot(x, bell_mf, label='Bell-Shaped MF')

plt.title('Fuzzy Membership Functions')
plt.xlabel('x')
plt.ylabel('Membership Degree')
plt.legend()
plt.grid(True)
plt.show()
```

Figure. 2: Fuzzy Membership Function Generated by the ChatGPT

- Prompt, make fuzzy membership with python and figure
- Chat GPT description, muzzy membership functions are used to represent fuzzy sets in fuzzy logic. They define the degree to which a given input belongs to a set. Here, we'll create some common types of fuzzy membership functions using Python and visualize them using Matplotlib. Types of Fuzzy Membership Functions; Triangular Membership Function; Trapezoidal Membership Function; Gaussian Membership Function; Bell-Shaped Membership Function; Code to Create and Visualize Fuzzy Membership Functions. First, we'll define the membership functions and then plot them. The end of ChatGPT description.
- Figure 2 shows the python code of fuzzy membership function generated by the ChatGPT. By using this code (copy), this paper generates the figure of this membership function as Figure 3.
- Explanation of ChatGPT, Triangular Membership Function: Defined by three points (a, b, c). The membership degree is 1 at point b and 0 at points a and c. Trapezoidal Membership Function: Defined by four points (a, b, c, d). The membership degree is 1 between points b and c and 0 at points a and d. Gaussian Membership Function: Defined by a mean and a standard deviation (sigma). The membership degree is 1 at the mean and decreases symmetrically. Bell-Shaped Membership Function: Defined by three parameters (a, b, c) where a control the width, b controls the slope, and c is the center.

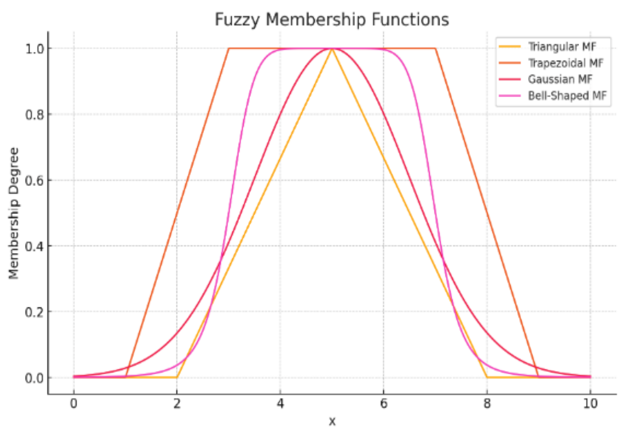

Figure. 3: The Fuzzy Membership Graph Generated by ChatGPT

Running the code above will generate a plot with the four membership functions, allowing you to visualize how different inputs map to their membership degrees in fuzzy logic. Here is the plot showing the different fuzzy membership functions (Figure 3):

- Triangular Membership Function (Triangular MF)
- Trapezoidal Membership Function (Trapezoidal MF)
- Gaussian Membership Function (Gaussian MF)
- Bell-Shaped Membership Function (Bell-Shaped MF)

Each curve represents how the membership degree varies with the input value xxx

Retrieval Number: 100.1/ijrte.C813913030924
DOI: 10.35940/ijrte.C8139.13030924
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

B. Anaconda Experiment

To prove the ChatGPT results, this paper experiment the Python fuzzy membership code generated by ChatGPT on Anaconda and tries to obtain the graph. Figure 4 is Anaconda window of Figure 1 and Figure 4-6 show the membership shape by Anaconda.

```python
import numpy as np
import matplotlib.pyplot as plt

def triangular_mf(x, a, b, c):
    return np.maximum(0, np.minimum((x - a) / (b - a), (c - x) / (c - b)))

# Example usage
x = np.linspace(0, 10, 1000)
a, b, c = 2, 5, 8
y = triangular_mf(x, a, b, c)

plt.plot(x, y)
plt.title("Triangular Membership Function")
plt.xlabel("x")
plt.ylabel("Membership degree")
plt.grid(True)
plt.show()
```

Figure. 4: (a) Triangular Membership Code of Anaconda

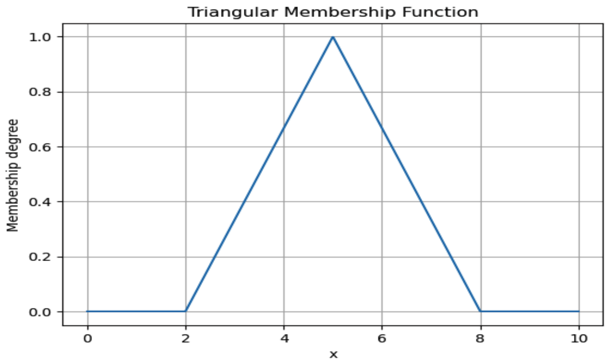

Figure. 4: (b) Triangular Membership Shape of Anaconda

```python
def trapezoidal_mf(x, a, b, c, d):
    return np.maximum(0, np.minimum(np.minimum((x - a) / (b - a), 1), (d - x) / (d - c)))

# Example usage
x = np.linspace(0, 10, 1000)
a, b, c, d = 2, 4, 6, 8
y = trapezoidal_mf(x, a, b, c, d)

plt.plot(x, y)
plt.title("Trapezoidal Membership Function")
plt.xlabel("x")
plt.ylabel("Membership degree")
plt.grid(True)
plt.show()
```

Figure. 4 (b): Trapezoidal Membership Code of Anaconda

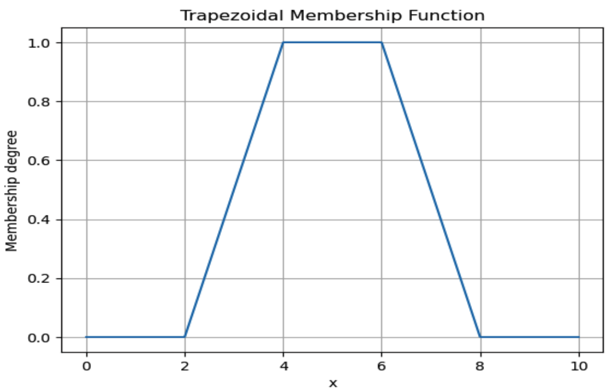

Figure. 4: (c).Trapezoidal Membership shape of Anaconda

```python
def gaussian_mf(x, mean, sigma):
    return np.exp(-0.5 * ((x - mean) / sigma) ** 2)

# Example usage
x = np.linspace(0, 10, 1000)
mean, sigma = 5, 1.5
y = gaussian_mf(x, mean, sigma)

plt.plot(x, y)
plt.title("Gaussian Membership Function")
plt.xlabel("x")
plt.ylabel("Membership degree")
plt.grid(True)
plt.show()
```

Figure. 5: (a) Gaussian Membership Code of Anaconda

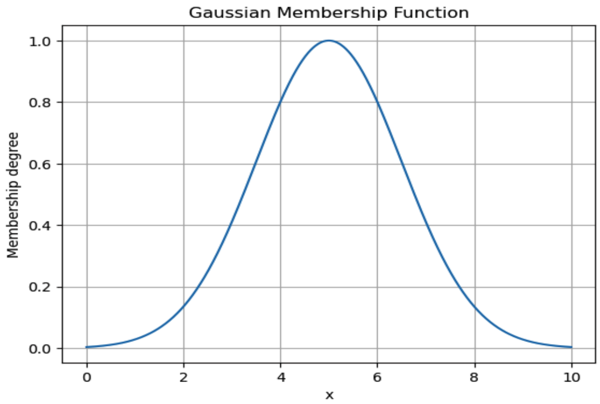

Figure. 5: (b) Gaussian Membership Shape of Anaconda

```python
def gbell_mf(x, a, b, c):
    return 1 / (1 + np.abs((x - c) / a) ** (2 * b))

# Example usage
x = np.linspace(0, 10, 1000)
a, b, c = 2, 4, 5
y = gbell_mf(x, a, b, c)

plt.plot(x, y)
plt.title("Generalized Bell Membership Function")
plt.xlabel("x")
plt.ylabel("Membership degree")
plt.grid(True)
plt.show()
```

Figure. 6: (a) Bell Membership Code of Anaconda

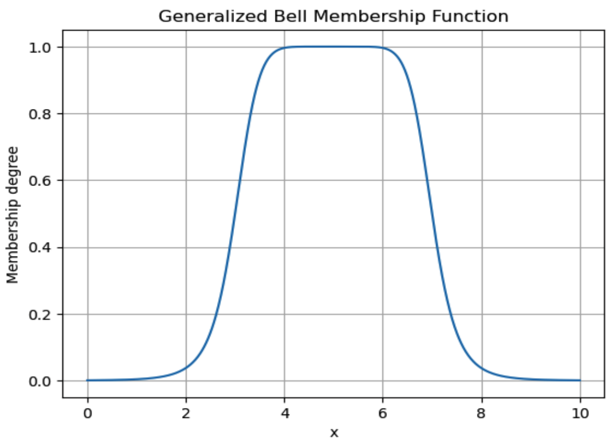

Figure. 6: (b) Bell Membership Shape of Anaconda

Retrieval Number: 100.1/ijrte.C813913030924
DOI: [10.35940/ijrte.C8139.13030924](https://doi.org/10.35940/ijrte.C8139.13030924)
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

C. Temperature Fuzzy Controller Design

ChatGPT description, the uploaded document appears to contain information related to fuzzy controllers. Let's break down the explanation and design a fuzzy controller based on the details provided.

- Explanation of the Figure
- The figure in the document likely represents a fuzzy membership function or a fuzzy inference system. Common elements in such figures include:
- Input Variables: These are the variables that the fuzzy controller takes as inputs. They can be things like temperature, pressure, speed, etc.
- Fuzzy Membership Functions: These functions map input values to their corresponding degree of membership in a fuzzy set. Typical membership functions include triangular, trapezoidal, Gaussian, and bell-shaped functions.
- Fuzzy Rules: These are if-then rules that describe how the fuzzy inputs relate to the fuzzy outputs. For example, "If temperature is high, then fan speed should be high."
- Defuzzification: This process converts the fuzzy output of the inference system into a crisp value. Common defuzzification methods include the centroid method and the max-membership method.
- To design a fuzzy controller, follow these steps:
- Define Input and Output Variables: Determine the inputs to the fuzzy controller and the desired outputs.
- Create Membership Functions: Define membership functions for each input and output variable.
- Establish Fuzzy Rules: Create a set of rules that define the relationship between input variables and output variables.
- Fuzzification: Convert the crisp input values into fuzzy values using the membership functions.
- Inference: Apply the fuzzy rules to the fuzzified inputs to determine the fuzzy outputs.
- Defuzzification: Convert the fuzzy outputs back into crisp values.
- Here is an example of designing a simple fuzzy controller in Python:
- Example: Temperature Control System, Step 1: Define Input and Output Variables, Input: Temperature (degrees Celsius), Output: Fan Speed (percentage)
- Step 2: Create Membership Functions

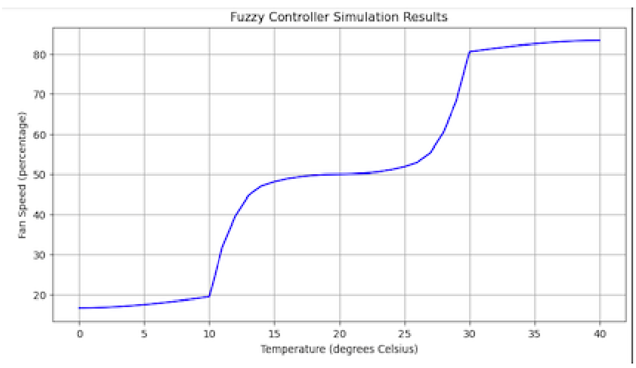

Figure. 7: The Simulation Result by ChatGPT Fuzzy Controller

Step 2: Create Membership Functions

```python
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt

# Define the universe of discourse for temperature and fan speed
temperature = np.arange(0, 41, 1)  # 0 to 40 degrees Celsius
fan_speed = np.arange(0, 101, 1)   # 0 to 100 percent

# Define membership functions for temperature
temp_low = fuzz.trimf(temperature, [0, 0, 20])
temp_medium = fuzz.trimf(temperature, [10, 20, 30])
temp_high = fuzz.trimf(temperature, [20, 40, 40])

# Define membership functions for fan speed
fan_low = fuzz.trimf(fan_speed, [0, 0, 50])
fan_medium = fuzz.trimf(fan_speed, [0, 50, 100])
fan_high = fuzz.trimf(fan_speed, [50, 100, 100])

# Plotting the membership functions
fig, (ax0, ax1) = plt.subplots(nrows=2, figsize=(8, 6))

ax0.plot(temperature, temp_low, 'b', linewidth=1.5, label='Low')
ax0.plot(temperature, temp_medium, 'g', linewidth=1.5, label='Medium')
ax0.plot(temperature, temp_high, 'r', linewidth=1.5, label='High')
ax0.set_title('Temperature')
ax0.legend()

ax1.plot(fan_speed, fan_low, 'b', linewidth=1.5, label='Low')
ax1.plot(fan_speed, fan_medium, 'g', linewidth=1.5, label='Medium')
ax1.plot(fan_speed, fan_high, 'r', linewidth=1.5, label='High')
ax1.set_title('Fan Speed')
ax1.legend()

plt.tight_layout()
plt.show()
```

Step 3: Establish Fuzzy Rules

```python
# Define the fuzzy rules
rule1 = fuzz.Rule(temp_low, fan_low)
rule2 = fuzz.Rule(temp_medium, fan_medium)
rule3 = fuzz.Rule(temp_high, fan_high)
```

Step 4: Fuzzification, Inference, and Defuzzification

```python
# Define the control system
fan_control = fuzz.ControlSystem([rule1, rule2, rule3])
fan_sim = fuzz.ControlSystemSimulation(fan_control)

# Provide a temperature input
fan_sim.input['temperature'] = 25

# Perform the defuzzification
fan_sim.compute()

# Get the fan speed output
print(fan_sim.output['fan_speed'])
```

Figure. 8: The Python Code Generated by ChatGPT

ChatGPT Description about Python code, this example outlines the basic process of creating a fuzzy controller using Python and the scikit-fuzzy library. The specific details and membership functions can be adjusted based on the requirements of your application. If you have specific details or further context from the document, I can incorporate those into the fuzzy controller design. Please let me know how you'd like to proceed!

- Prompt, show the results of simulation of this fuzzy controller
- ChatGPT Description, it seems there was an internal error while trying to simulate the fuzzy controller. Let's try to troubleshoot this step-by-step. Revised Approach for Fuzzy Controller Simulation
- I will re-implement the fuzzy controller simulation in smaller steps to identify and resolve the issue.

Retrieval Number: 100.1/ijrte.C813913030924
DOI: 10.35940/ijrte.C8139.13030924
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

- Step 1, define Membership Functions and Fuzzy Variables, First, let's redefine the membership functions and fuzzy variables.

```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for temperature and fan speed
temperature = np.arange(0, 41, 1)  # 0 to 40 degrees Celsius
fan_speed = np.arange(0, 101, 1)   # 0 to 100 percent

# Define membership functions for temperature
temp_low = fuzz.trimf(temperature, [0, 0, 20])
temp_medium = fuzz.trimf(temperature, [10, 20, 30])
temp_high = fuzz.trimf(temperature, [20, 40, 40])

# Define membership functions for fan speed
fan_low = fuzz.trimf(fan_speed, [0, 0, 50])
fan_medium = fuzz.trimf(fan_speed, [0, 50, 100])
fan_high = fuzz.trimf(fan_speed, [50, 100, 100])

# Define the fuzzy variables
temp = ctrl.Antecedent(temperature, 'temperature')
fan = ctrl.Consequent(fan_speed, 'fan_speed')

# Auto-membership function population is possible with .automf(3, 5, or 7)
temp['low'] = temp_low
temp['medium'] = temp_medium
temp['high'] = temp_high

fan['low'] = fan_low
fan['medium'] = fan_medium
fan['high'] = fan_high
```

Figure. 9: (a) The Python Code of Fuzzy Controller Generated by ChatGPT

Step 2: Define Fuzzy Rules and Control System

```python
# Define the fuzzy rules
rule1 = ctrl.Rule(temp['low'], fan['low'])
rule2 = ctrl.Rule(temp['medium'], fan['medium'])
rule3 = ctrl.Rule(temp['high'], fan['high'])

# Define the control system
fan_control = ctrl.ControlSystem([rule1, rule2, rule3])
fan_sim = ctrl.ControlSystemSimulation(fan_control)
```

Step 3: Perform Simulations and Plot Results

```python
# Perform simulations for a range of temperature values
temperature_values = np.arange(0, 41, 1)
fan_speed_results = []

for temp_value in temperature_values:
    fan_sim.input['temperature'] = temp_value
    fan_sim.compute()
    fan_speed_results.append(fan_sim.output['fan_speed'])

# Plotting the results
plt.figure(figsize=(10, 5))
plt.plot(temperature_values, fan_speed_results, 'b', linewidth=1.5)
plt.title('Fuzzy Controller Simulation Results')
plt.xlabel('Temperature (degrees Celsius)')
plt.ylabel('Fan Speed (percentage)')
plt.grid(True)
plt.show()
```

Figure. 9: (b) The Python Code of Fuzzy Controller Generated by ChatGPT

I'll run the code again to ensure it executes correctly.

- Prompt, why it has problem import skfuzzy as fuzz
- ChatGPT Description, the problem might be caused by the runtime environment not having the necessary libraries or not supporting the execution of certain operations. Let's ensure the fuzzy logic simulation is correct and explain the expected process step by step.
- To simulate a fuzzy controller, you should install the scikit-fuzzy library. If you are running this locally, make sure to install the library using the following command:

```sh
pip install scikit-fuzzy
```

Here's the entire process again in one script:

```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for temperature and fan speed
temperature = np.arange(0, 41, 1)  # 0 to 40 degrees Celsius
fan_speed = np.arange(0, 101, 1)   # 0 to 100 percent

# Define membership functions for temperature
temp_low = fuzz.trimf(temperature, [0, 0, 20])
temp_medium = fuzz.trimf(temperature, [10, 20, 30])
temp_high = fuzz.trimf(temperature, [20, 40, 40])

# Define membership functions for fan speed
fan_low = fuzz.trimf(fan_speed, [0, 0, 50])
fan_medium = fuzz.trimf(fan_speed, [0, 50, 100])
fan_high = fuzz.trimf(fan_speed, [50, 100, 100])

# Define the fuzzy variables
temp = ctrl.Antecedent(temperature, 'temperature')
fan = ctrl.Consequent(fan_speed, 'fan_speed')

# Auto-membership function population is possible with .automf(3, 5, or 7)
temp['low'] = temp_low
temp['medium'] = temp_medium
temp['high'] = temp_high

fan['low'] = fan_low
fan['medium'] = fan_medium
fan['high'] = fan_high

# Define the fuzzy rules
rule1 = ctrl.Rule(temp['low'], fan['low'])
rule2 = ctrl.Rule(temp['medium'], fan['medium'])
rule3 = ctrl.Rule(temp['high'], fan['high'])

# Define the control system
fan_control = ctrl.ControlSystem([rule1, rule2, rule3])
fan_sim = ctrl.ControlSystemSimulation(fan_control)

# Perform simulations for a range of temperature values
temperature_values = np.arange(0, 41, 1)
fan_speed_results = []

for temp_value in temperature_values:
    fan_sim.input['temperature'] = temp_value
    fan_sim.compute()
    fan_speed_results.append(fan_sim.output['fan_speed'])

# Plotting the results
plt.figure(figsize=(10, 5))
plt.plot(temperature_values, fan_speed_results, 'b', linewidth=1.5)
plt.title('Fuzzy Controller Simulation Results')
plt.xlabel('Temperature (degrees Celsius)')
plt.ylabel('Fan Speed (percentage)')
plt.grid(True)
plt.show()
```

Figure. 10: The Python Code of Fuzzy Controller Generated by ChatGPT After Skfuzzy Coding

Retrieval Number: 100.1/ijrte.C813913030924
DOI: [10.35940/ijrte.C8139.13030924](https://doi.org/10.35940/ijrte.C8139.13030924)
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

- ChatGPT description, ensure all necessary libraries (numpy, skfuzzy, matplotlib) are imported.
- Define variables and membership functions, Define the range for temperature and fan speed, Create triangular membership functions for low, medium, and high values of both temperature and fan speed.
- Define Fuzzy Variables, use ctrl, antecedent for input variables and ctrl. Consequent for output variables.
- Define Fuzzy Rules, create rules that map the input variables to the output variables.
- Create Control System and Simulation, create a Control System and a Control System Simulation using the defined rules.
- Perform Simulations, Run the simulation for a range of temperature values and collect the fan speed results.
- Plot Results, use matplotlib to visualize the results.
- If you encounter any issues, make sure the scikit-fuzzy library is correctly installed and try running the script in a local Python environment. If the problem persists, there may be an issue specific to the runtime environment or the library versions.

```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for temperature and fan speed
temperature = np.arange(0, 41, 1)  # 0 to 40 degrees Celsius
fan_speed = np.arange(0, 101, 1)   # 0 to 100 percent

# Define membership functions for temperature
temp_low = fuzz.trimf(temperature, [0, 0, 20])
temp_medium = fuzz.trimf(temperature, [10, 20, 30])
temp_high = fuzz.trimf(temperature, [20, 40, 40])

# Define membership functions for fan speed
fan_low = fuzz.trimf(fan_speed, [0, 0, 50])
fan_medium = fuzz.trimf(fan_speed, [0, 50, 100])
fan_high = fuzz.trimf(fan_speed, [50, 100, 100])

# Define the fuzzy variables
temp = ctrl.Antecedent(temperature, 'temperature')
fan = ctrl.Consequent(fan_speed, 'fan_speed')

# Assign membership functions to fuzzy variables
temp['low'] = temp_low
temp['medium'] = temp_medium
temp['high'] = temp_high

fan['low'] = fan_low
fan['medium'] = fan_medium
fan['high'] = fan_high

# Define the fuzzy rules
rule1 = ctrl.Rule(temp['low'], fan['low'])
rule2 = ctrl.Rule(temp['medium'], fan['medium'])
rule3 = ctrl.Rule(temp['high'], fan['high'])

# Define the control system
fan_control = ctrl.ControlSystem([rule1, rule2, rule3])
fan_sim = ctrl.ControlSystemSimulation(fan_control)
```

Figure. 11: (a) The Python Code Corrected of Fuzzy Controller Generated by ChatGPT after Skfuzzy Coding

```python
# Perform simulations for a range of temperature values
temperature_values = np.arange(0, 41, 1)
fan_speed_results = []

for temp_value in temperature_values:
    fan_sim.input['temperature'] = temp_value
    fan_sim.compute()
    fan_speed_results.append(fan_sim.output['fan_speed'])

# Plotting the results
plt.figure(figsize=(10, 5))
plt.plot(temperature_values, fan_speed_results, 'b', linewidth=1.5)
plt.title('Fuzzy Controller Simulation Results')
plt.xlabel('Temperature (degrees Celsius)')
plt.ylabel('Fan Speed (percentage)')
plt.grid(True)
plt.show()
```

Figure. 11: (b) The Python Code Corrected of Fuzzy Controller Generated by ChatGPT After Skfuzzy Coding

- Prompt, correct import skfuzzy as fuzz
- ChatGPT Description, let's correct the script to ensure it works properly with the scikit-fuzzy library. Here is a step-by-step guide to simulate a fuzzy controller for a temperature control system, ensuring all necessary imports and functions are correctly defined.
- Define Membership Functions and Fuzzy Variables, here is the Python code to define the membership functions and fuzzy variables:
- ChatGPT Description, perform simulations and plot results, here is the code to perform simulations and plot the results, this code will create and simulate a fuzzy controller that adjusts the fan speed based on the temperature input. The results will be plotted to show how the fan speed changes with varying temperatures.
- If you encounter any issues, please ensure you have all the necessary libraries installed and that you are running the script in a compatible Python environment.

D. Anaconda Experiment of Temperature Fuzzy Controller Design

```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for temperature and fan speed
temperature = np.arange(0, 41, 1)  # 0 to 40 degrees Celsius
fan_speed = np.arange(0, 101, 1)   # 0 to 100 percent

# Define membership functions for temperature
temp_low = fuzz.trimf(temperature, [0, 0, 20])
temp_medium = fuzz.trimf(temperature, [10, 20, 30])
temp_high = fuzz.trimf(temperature, [20, 40, 40])

# Define membership functions for fan speed
fan_low = fuzz.trimf(fan_speed, [0, 0, 50])
fan_medium = fuzz.trimf(fan_speed, [0, 50, 100])
fan_high = fuzz.trimf(fan_speed, [50, 100, 100])

# Define the fuzzy variables
temp = ctrl.Antecedent(temperature, 'temperature')
fan = ctrl.Consequent(fan_speed, 'fan_speed')

# Auto-membership function population is possible with .automf(3, 5, or 7)
temp['low'] = temp_low
temp['medium'] = temp_medium
temp['high'] = temp_high

fan['low'] = fan_low
fan['medium'] = fan_medium
fan['high'] = fan_high

# Define the fuzzy rules
rule1 = ctrl.Rule(temp['low'], fan['low'])
rule2 = ctrl.Rule(temp['medium'], fan['medium'])
rule3 = ctrl.Rule(temp['high'], fan['high'])

# Define the control system
fan_control = ctrl.ControlSystem([rule1, rule2, rule3])
fan_sim = ctrl.ControlSystemSimulation(fan_control)

# Perform simulations for a range of temperature values
temperature_values = np.arange(0, 41, 1)
fan_speed_results = []

for temp_value in temperature_values:
    fan_sim.input['temperature'] = temp_value
    fan_sim.compute()
    fan_speed_results.append(fan_sim.output['fan_speed'])

# Plotting the results
plt.figure(figsize=(10, 5))
plt.plot(temperature_values, fan_speed_results, 'b', linewidth=1.5)
plt.title('Fuzzy Controller Simulation Results')
plt.xlabel('Temperature (degrees Celsius)')
plt.ylabel('Fan Speed (percentage)')
plt.grid(True)
plt.show()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[3], line 2
      1 import numpy as np
----> 2 import skfuzzy as fuzz
      3 from skfuzzy import control as ctrl
      4 import matplotlib.pyplot as plt

ModuleNotFoundError: No module named 'skfuzzy'
```

Retrieval Number: 100.1/ijrte.C813913030924
DOI: 10.35940/ijrte.C8139.13030924
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-13 Issue-3, September 2024

```
[4]: pip install scikit-fuzzy

Defaulting to user installation because normal site-packages is not writeable
Looking in links: /usr/share/pip-wheels
Collecting scikit-fuzzy
  Downloading scikit-fuzzy-0.4.2.tar.gz (993 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 994.0/994.0 kB 18.1 MB/s eta 0:00:000:00:01
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy>=1.6.0 in /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages (from scikit-fuzzy) (1.24.3)
Requirement already satisfied: scipy>=0.9.0 in /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages (from scikit-fuzzy) (1.11.1)
Requirement already satisfied: networkx>=1.9.0 in /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages (from scikit-fuzzy) (3.1)
Building wheels for collected packages: scikit-fuzzy
  Building wheel for scikit-fuzzy (setup.py) ... done
  Created wheel for scikit-fuzzy: filename=scikit_fuzzy-0.4.2-py3-none-any.whl size=894072 sha256=81335b603bd44d1e15604831e53b6d259885cfcbd60a21352699f34c98885118
  Stored in directory: /home/a2ab43b4-0c6a-486f-879c-46ca306f2930/.cache/pip/wheels/b1/67/d1/6c29aeb1cd2d36447ad5b0de027580b8894aff702a9b50041e
Successfully built scikit-fuzzy
Installing collected packages: scikit-fuzzy
Successfully installed scikit-fuzzy-0.4.2
Note: you may need to restart the kernel to use updated packages.
```

```
[5]: import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for temperature and fan speed
temperature = np.arange(0, 41, 1)  # 0 to 40 degrees Celsius
fan_speed = np.arange(0, 101, 1)   # 0 to 100 percent

# Define membership functions for temperature
temp_low = fuzz.trimf(temperature, [0, 0, 20])
temp_medium = fuzz.trimf(temperature, [10, 20, 30])
temp_high = fuzz.trimf(temperature, [20, 40, 40])

# Define membership functions for fan speed
fan_low = fuzz.trimf(fan_speed, [0, 0, 50])
fan_medium = fuzz.trimf(fan_speed, [0, 50, 100])
fan_high = fuzz.trimf(fan_speed, [50, 100, 100])

# Define the fuzzy variables
temp = ctrl.Antecedent(temperature, 'temperature')
fan = ctrl.Consequent(fan_speed, 'fan_speed')

# Assign membership functions to fuzzy variables
temp['low'] = temp_low
temp['medium'] = temp_medium
temp['high'] = temp_high

fan['low'] = fan_low
fan['medium'] = fan_medium
fan['high'] = fan_high

# Define the fuzzy rules
rule1 = ctrl.Rule(temp['low'], fan['low'])
rule2 = ctrl.Rule(temp['medium'], fan['medium'])
rule3 = ctrl.Rule(temp['high'], fan['high'])

# Define the control system
fan_control = ctrl.ControlSystem([rule1, rule2, rule3])
fan_sim = ctrl.ControlSystemSimulation(fan_control)
```

```
[6]: # Perform simulations for a range of temperature values
temperature_values = np.arange(0, 41, 1)
fan_speed_results = []

for temp_value in temperature_values:
    fan_sim.input['temperature'] = temp_value
    fan_sim.compute()
    fan_speed_results.append(fan_sim.output['fan_speed'])

# Plotting the results
plt.figure(figsize=(10, 5))
plt.plot(temperature_values, fan_speed_results, 'b', linewidth=1.5)
plt.title('Fuzzy Controller Simulation Results')
plt.xlabel('Temperature (degrees Celsius)')
plt.ylabel('Fan Speed (percentage)')
plt.grid(True)
plt.show()
```

Figure. 12: Temperature Fuzzy Controller Designed by Anaconda

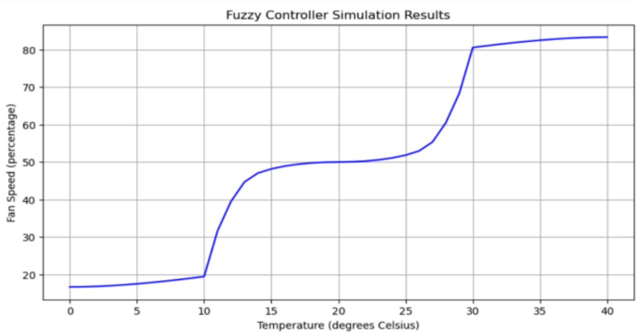

Figure. 13: Temperature Fuzzy Controller Simulation Result Obtained by Anaconda

E. Pendulum Fuzzy Controller Design by ChatGPT

This paper designs the fuzzy controller of pendulum by using ChatGPT. Figure 13 presents the block diagram uploaded for ChatGPT. ChatGPT read this MS work block diagram and generates many Python code and simulation results.

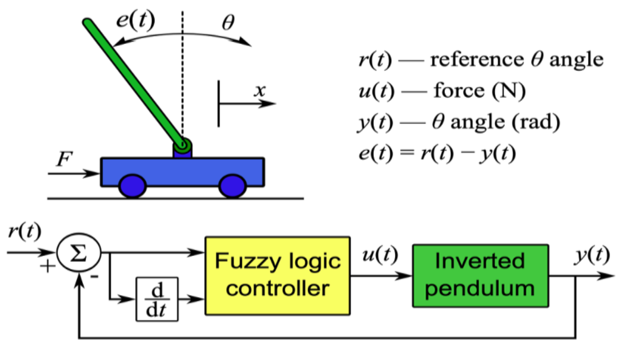

Figure. 14: The Pendulum Fuzzy Control System Upload in ChatGPT

- ChatGPT Description: It appears we are working with a document that includes the design for a pendulum fuzzy controller. Let's explain the steps to design such a controller and simulate its behavior using Python.

```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for angle, angular velocity, and control force
angle = np.arange(-180, 181, 1)         # Angle from -180 to 180 degrees
angular_velocity = np.arange(-10, 11, 1)  # Angular velocity from -10 to 10
control_force = np.arange(-100, 101, 1)   # Control force from -100 to 100

# Define membership functions for angle
angle_negative_large = fuzz.trimf(angle, [-180, -180, -90])
angle_negative_medium = fuzz.trimf(angle, [-180, -90, 0])
angle_zero = fuzz.trimf(angle, [-90, 0, 90])
angle_positive_medium = fuzz.trimf(angle, [0, 90, 180])
angle_positive_large = fuzz.trimf(angle, [90, 180, 180])

# Define membership functions for angular velocity
ang_vel_negative = fuzz.trimf(angular_velocity, [-10, -10, 0])
ang_vel_zero = fuzz.trimf(angular_velocity, [-10, 0, 10])
ang_vel_positive = fuzz.trimf(angular_velocity, [0, 10, 10])

# Define membership functions for control force
force_negative = fuzz.trimf(control_force, [-100, -100, 0])
force_zero = fuzz.trimf(control_force, [-100, 0, 100])
force_positive = fuzz.trimf(control_force, [0, 100, 100])

# Define the fuzzy variables
angle_var = ctrl.Antecedent(angle, 'angle')
angular_velocity_var = ctrl.Antecedent(angular_velocity, 'angular_velocity')
control_force_var = ctrl.Consequent(control_force, 'control_force')

# Assign membership functions to fuzzy variables
angle_var['negative_large'] = angle_negative_large
angle_var['negative_medium'] = angle_negative_medium
angle_var['zero'] = angle_zero
angle_var['positive_medium'] = angle_positive_medium
angle_var['positive_large'] = angle_positive_large

angular_velocity_var['negative'] = ang_vel_negative
angular_velocity_var['zero'] = ang_vel_zero
angular_velocity_var['positive'] = ang_vel_positive

control_force_var['negative'] = force_negative
control_force_var['zero'] = force_zero
control_force_var['positive'] = force_positive
```

Figure. 15: (a) The Pendulum Fuzzy Control Python Code Generated by ChatGPT

```python
# Define the fuzzy rules
rule1 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['negative'],
rule2 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['zero'], cont
rule3 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['positive'],

rule4 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['negative'],
rule5 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['zero'], con
rule6 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['positive'],

rule7 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['negative'], control_fo
rule8 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['zero'], control_force_
rule9 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['positive'], control_fo

rule10 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['negative']
rule11 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['zero'], co
rule12 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['positive']

rule13 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['negative'],
rule14 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['zero'], con
rule15 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['positive'],
```

Figure: 15. (b) The Pendulum Fuzzy Control Python Code Generated by ChatGPT

- Step-by-Step Design and Simulation of a Pendulum Fuzzy Controller, 1. Define Membership Functions and Fuzzy Variables, we will create membership functions for the input variables (angle and angular velocity) and the output variable (control force), 2. Define Fuzzy Rules, we will set up rules that describe the behavior of the pendulum based on the fuzzy logic, Create Control System and Simulation, we will implement the control system using the fuzzy rules and simulate its performance[18][19][20][21][22].
- Implementation, let's start with the code to define the membership functions, rules, and simulate the behavior of the pendulum fuzzy controller.
- Import Libraries, First, ensure you have scikit-fuzzy installed:
- Create Control System and Simulation

```python
# Define the control system
pendulum_control = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, r
pendulum_sim = ctrl.ControlSystemSimulation(pendulum_control)

# Simulate the controller
angles = np.linspace(-180, 180, 361)
ang_velocities = np.linspace(-10, 10, 21)
force_results = np.zeros((len(angles), len(ang_velocities)))

for i, angle_value in enumerate(angles):
    for j, ang_vel_value in enumerate(ang_velocities):
        pendulum_sim.input['angle'] = angle_value
        pendulum_sim.input['angular_velocity'] = ang_vel_value
        pendulum_sim.compute()
        force_results[i, j] = pendulum_sim.output['control_force']

# Plotting the results
X, Y = np.meshgrid(angles, ang_velocities)
Z = force_results.T

plt.figure(figsize=(10, 8))
cp = plt.contourf(X, Y, Z, cmap='coolwarm')
plt.colorbar(cp)
plt.title('Fuzzy Pendulum Control Surface')
plt.xlabel('Angle (degrees)')
plt.ylabel('Angular Velocity')
plt.show()
```

Figure. 15: (c) The Pendulum Fuzzy Control Python Code Generated by ChatGPT

- ChatGPT Description, this script defines the membership functions, sets up the fuzzy rules, creates a control system, simulates the controller for a range of angles and angular velocities, and finally plots the results. The plot shows the control force as a function of the angle and angular velocity of the pendulum.

F. Experiment of Pendulum Fuzzy Controller Design on Anaconda

```python
[8]:
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for angle, angular velocity, and control force
angle = np.arange(-180, 181, 1)        # Angle from -180 to 180 degrees
angular_velocity = np.arange(-10, 11, 1)  # Angular velocity from -10 to 10
control_force = np.arange(-100, 101, 1)   # Control force from -100 to 100

# Define membership functions for angle
angle_negative_large = fuzz.trimf(angle, [-180, -180, -90])
angle_negative_medium = fuzz.trimf(angle, [-180, -90, 0])
angle_zero = fuzz.trimf(angle, [-90, 0, 90])
angle_positive_medium = fuzz.trimf(angle, [0, 90, 180])
angle_positive_large = fuzz.trimf(angle, [90, 180, 180])

# Define membership functions for angular velocity
ang_vel_negative = fuzz.trimf(angular_velocity, [-10, -10, 0])
ang_vel_zero = fuzz.trimf(angular_velocity, [-10, 0, 10])
ang_vel_positive = fuzz.trimf(angular_velocity, [0, 10, 10])

# Define membership functions for control force
force_negative = fuzz.trimf(control_force, [-100, -100, 0])
force_zero = fuzz.trimf(control_force, [-100, 0, 100])
force_positive = fuzz.trimf(control_force, [0, 100, 100])

# Define the fuzzy variables
angle_var = ctrl.Antecedent(angle, 'angle')
angular_velocity_var = ctrl.Antecedent(angular_velocity, 'angular_velocity')
control_force_var = ctrl.Consequent(control_force, 'control_force')

# Assign membership functions to fuzzy variables
angle_var['negative_large'] = angle_negative_large
angle_var['negative_medium'] = angle_negative_medium
angle_var['zero'] = angle_zero
angle_var['positive_medium'] = angle_positive_medium
angle_var['positive_large'] = angle_positive_large

angular_velocity_var['negative'] = ang_vel_negative
angular_velocity_var['zero'] = ang_vel_zero
angular_velocity_var['positive'] = ang_vel_positive

control_force_var['negative'] = force_negative
control_force_var['zero'] = force_zero
control_force_var['positive'] = force_positive
```

```python
[9]:
# Define the fuzzy rules
rule1 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['negative'], control_force_var['positive'])
rule2 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['zero'], control_force_var['positive'])
rule3 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['positive'], control_force_var['zero'])

rule4 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['negative'], control_force_var['positive'])
rule5 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['zero'], control_force_var['positive'])
rule6 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['positive'], control_force_var['zero'])

rule7 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['negative'], control_force_var['positive'])
rule8 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['zero'], control_force_var['zero'])
rule9 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['positive'], control_force_var['negative'])

rule10 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['negative'], control_force_var['zero'])
rule11 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['zero'], control_force_var['negative'])
rule12 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['positive'], control_force_var['negative'])

rule13 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['negative'], control_force_var['zero'])
rule14 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['zero'], control_force_var['negative'])
rule15 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['positive'], control_force_var['negative'])
```

```python
[10]:
# Define the control system
pendulum_control = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9, rule10, rule11, rule12, rule13, rule14, rule15])
pendulum_sim = ctrl.ControlSystemSimulation(pendulum_control)

# Simulate the controller
angles = np.linspace(-180, 180, 361)
ang_velocities = np.linspace(-10, 10, 21)
force_results = np.zeros((len(angles), len(ang_velocities)))

for i, angle_value in enumerate(angles):
    for j, ang_vel_value in enumerate(ang_velocities):
        pendulum_sim.input['angle'] = angle_value
        pendulum_sim.input['angular_velocity'] = ang_vel_value
        pendulum_sim.compute()
        force_results[i, j] = pendulum_sim.output['control_force']

# Plotting the results
X, Y = np.meshgrid(angles, ang_velocities)
Z = force_results.T

plt.figure(figsize=(10, 8))
cp = plt.contourf(X, Y, Z, cmap='coolwarm')
plt.colorbar(cp)
plt.title('Fuzzy Pendulum Control Surface')
plt.xlabel('Angle (degrees)')
plt.ylabel('Angular Velocity')
plt.show()
```

Figure. 16: The Pendulum Fuzzy Control Python Code Generated by Anaconda

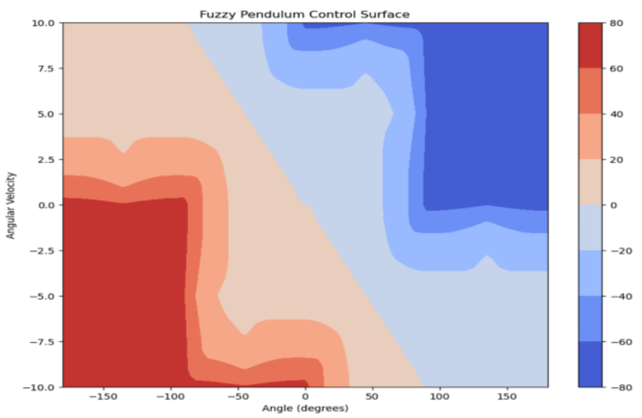

Figure. 17: The Pendulum Fuzzy Control Result Generated by Anaconda

Retrieval Number: 100.1/ijrte.C813913030924
DOI: 10.35940/ijrte.C8139.13030924
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

- Prompt, explain this figure at pendulum simulation you design
- ChatGPT description, the figure shows the control surface of a fuzzy controller designed for a pendulum system, depicting the control force as a function of the pendulum's angle and angular velocity. Let's break down the key components and what they represent:
- X-Axis (Angle in degrees): Represents the angle of the pendulum, ranging from -180 degrees (pendulum leaning far left) to +180 degrees (pendulum leaning far right).
- Y-Axis (Angular Velocity): Represents the angular velocity of the pendulum, ranging from -10 (pendulum moving counterclockwise) to +10 (pendulum moving clockwise).
- Color Scale: Represents the control force applied to the pendulum. The scale ranges from -80 (deep blue) to +80 (deep red).
- Negative Values (Blue Shades): Indicate a corrective force applied to move the pendulum to the left.
- Positive Values (Red Shades): Indicate a corrective force applied to move the pendulum to the right.
- Zero (Neutral Colors): No corrective force is applied; the pendulum is in balance.
- Central Region (Near Zero Angle and Angular Velocity), Here, the control force is minimal (near zero), suggesting that the pendulum is close to its upright position and stable, requiring little to no corrective force.
- Regions with High Angle and Low Angular Velocity, Top Right and Bottom Left Corners: These regions show significant control forces (deep red and blue) because the pendulum is far from its stable position and needs a strong corrective force to return to the upright position.
- Transition Zones, Sloped Regions Between Colors: The gradual change from blue to red indicates the fuzzy controller's smooth transition in the control force as the pendulum's angle and velocity change.
- Functionality:
- The controller aims to stabilize the pendulum by applying appropriate forces depending on its angle and angular velocity. For instance, When the pendulum has a large positive angle (leaning far to the right), a strong negative force (deep blue) is applied to push it back.
- When the pendulum is moving quickly (high angular velocity) in either direction, the controller applies a force to counteract this movement and bring the pendulum back to rest.
- Stability, the figure illustrates how the fuzzy controller manages to keep the pendulum stable. When the pendulum is near its stable position (angle near zero), the applied forces are minimal. As the pendulum moves further from the stable position, the applied forces increase, demonstrating the controller's effectiveness in stabilizing the pendulum.
- Smooth Control, the smooth transitions in the control force suggest that the fuzzy controller provides a continuous and gradual adjustment, which is crucial for maintaining the pendulum's stability without causing oscillations or abrupt movements. Overall, this control surface provides a visual understanding of how the fuzzy controller adjusts the force based on the pendulum's state, ensuring stability and smooth operation.
- Prompt, show the control curve for this pendulum behavior
- ChatGPT description, to show the control curve for the pendulum behavior, we will simulate the pendulum's motion over time and plot the control force applied by the fuzzy controller at each step. We'll create a simple simulation where the pendulum starts with an initial angle and angular velocity, and the fuzzy controller adjusts the force to stabilize it.
- Define Initial Conditions, set initial angle and angular velocity.
- Apply Fuzzy Controller, at each time step, compute the control force using the fuzzy controller.
- Update Pendulum State, update the angle and angular velocity based on the control force.
- Plot Results, plot the control force and the pendulum's angle over time.
- Define the Pendulum Dynamics and Simulation

python 코드 복사

```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for angle, angular velocity, and control force
angle = np.arange(-180, 181, 1)       # Angle from -180 to 180 degrees
angular_velocity = np.arange(-10, 11, 1)  # Angular velocity from -10 to 10
control_force = np.arange(-100, 101, 1)   # Control force from -100 to 100

# Define membership functions for angle
angle_negative_large = fuzz.trimf(angle, [-180, -180, -90])
angle_negative_medium = fuzz.trimf(angle, [-180, -90, 0])
angle_zero = fuzz.trimf(angle, [-90, 0, 90])
angle_positive_medium = fuzz.trimf(angle, [0, 90, 180])
angle_positive_large = fuzz.trimf(angle, [90, 180, 180])

# Define membership functions for angular velocity
ang_vel_negative = fuzz.trimf(angular_velocity, [-10, -10, 0])
ang_vel_zero = fuzz.trimf(angular_velocity, [-10, 0, 10])
ang_vel_positive = fuzz.trimf(angular_velocity, [0, 10, 10])

# Define membership functions for control force
force_negative = fuzz.trimf(control_force, [-100, -100, 0])
force_zero = fuzz.trimf(control_force, [-100, 0, 100])
force_positive = fuzz.trimf(control_force, [0, 100, 100])

# Define the fuzzy variables
angle_var = ctrl.Antecedent(angle, 'angle')
angular_velocity_var = ctrl.Antecedent(angular_velocity, 'angular_velocity')
control_force_var = ctrl.Consequent(control_force, 'control_force')

# Assign membership functions to fuzzy variables
angle_var['negative_large'] = angle_negative_large
angle_var['negative_medium'] = angle_negative_medium
angle_var['zero'] = angle_zero
angle_var['positive_medium'] = angle_positive_medium
angle_var['positive_large'] = angle_positive_large

angular_velocity_var['negative'] = ang_vel_negative
angular_velocity_var['zero'] = ang_vel_zero
angular_velocity_var['positive'] = ang_vel_positive

control_force_var['negative'] = force_negative
control_force_var['zero'] = force_zero
control_force_var['positive'] = force_positive
```

Figure. 18: (a) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Curve Prompt

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

```
# Define the fuzzy rules
rule1 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['negative'],
rule2 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['zero'], cont
rule3 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['positive'],

rule4 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['negative'],
rule5 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['zero'], con
rule6 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['positive'],

rule7 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['negative'], control_fo
rule8 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['zero'], control_force_
rule9 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['positive'], control_fo

rule10 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['negative']
rule11 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['zero'], co
rule12 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['positive']

rule13 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['negative'],
rule14 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['zero'], con
rule15 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['positive'],

# Define the control system
pendulum_control = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, r
pendulum_sim = ctrl.ControlSystemSimulation(pendulum_control)

# Pendulum dynamics
def update_pendulum_state(angle, ang_vel, force, dt):
    # Constants for the pendulum (assumed for simplicity)
    length = 1.0  # length of the pendulum
    mass = 1.0    # mass of the pendulum
    g = 9.81      # acceleration due to gravity

    # Equations of motion (simplified)
    ang_acc = (force - mass * g * np.sin(np.radians(angle))) / (mass * length)

    # Update state
    ang_vel += ang_acc * dt
    angle += ang_vel * dt

    return angle, ang_vel
```

Figure. 18: (b) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Curve Prompt

```
# Initial conditions
initial_angle = 90
initial_ang_vel = 0
dt = 0.1  # time step
time_steps = 100

angles = []
ang_vels = []
forces = []

angle = initial_angle
ang_vel = initial_ang_vel

# Simulation loop
for _ in range(time_steps):
    pendulum_sim.input['angle'] = angle
    pendulum_sim.input['angular_velocity'] = ang_vel
    pendulum_sim.compute()
    force = pendulum_sim.output['control_force']

    angle, ang_vel = update_pendulum_state(angle, ang_vel, force, dt)

    angles.append(angle)
    ang_vels.append(ang_vel)
    forces.append(force)

# Plotting the results
time = np.arange(0, time_steps * dt, dt)

plt.figure(figsize=(14, 7))

plt.subplot(2, 1, 1)
plt.plot(time, angles, label='Angle (degrees)')
plt.plot(time, ang_vels, label='Angular Velocity')
plt.title('Pendulum State Over Time')
plt.xlabel('Time (s)')
plt.ylabel('State')
plt.legend()
plt.grid()

plt.subplot(2, 1, 2)
plt.plot(time, forces, label='Control Force', color='r')
plt.title('Control Force Over Time')
plt.xlabel('Time (s)')
plt.ylabel('Force')
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
```

Figure. 18: (c) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Curve Prompt

ChatGPT Description

Pendulum Dynamics, the update_pendulum_state function updates the angle and angular velocity of the pendulum based on the applied control force and the equations of motion.

Simulation Loop, the loop iterates over the specified number of time steps. At each step, the fuzzy controller computes the control force based on the current angle and angular velocity. The pendulum state (angle and angular velocity) is updated using the computed control force.

The angle, angular velocity, and control force are stored for plotting.

Running this code will simulate the behavior of the pendulum under the control of the fuzzy controller and produce plots showing the state of the pendulum and the control force over time.

Prompt, can you show simulation curve?

ChatGPT Description, let's simulate the fuzzy controller for the pendulum and display the resulting curves for the pendulum's angle, angular velocity, and the control force over time. Here's the complete code to achieve this:

```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for angle, angular velocity, and control fo
angle = np.arange(-180, 181, 1)      # Angle from -180 to 180 degrees
angular_velocity = np.arange(-10, 11, 1)  # Angular velocity from -10 to 10
control_force = np.arange(-100, 101, 1)  # Control force from -100 to 100

# Define membership functions for angle
angle_negative_large = fuzz.trimf(angle, [-180, -180, -90])
angle_negative_medium = fuzz.trimf(angle, [-180, -90, 0])
angle_zero = fuzz.trimf(angle, [-90, 0, 90])
angle_positive_medium = fuzz.trimf(angle, [0, 90, 180])
angle_positive_large = fuzz.trimf(angle, [90, 180, 180])

# Define membership functions for angular velocity
ang_vel_negative = fuzz.trimf(angular_velocity, [-10, -10, 0])
ang_vel_zero = fuzz.trimf(angular_velocity, [-10, 0, 10])
ang_vel_positive = fuzz.trimf(angular_velocity, [0, 10, 10])

# Define membership functions for control force
force_negative = fuzz.trimf(control_force, [-100, -100, 0])
force_zero = fuzz.trimf(control_force, [-100, 0, 100])
force_positive = fuzz.trimf(control_force, [0, 100, 100])

# Define the fuzzy variables
angle_var = ctrl.Antecedent(angle, 'angle')
angular_velocity_var = ctrl.Antecedent(angular_velocity, 'angular_velocity')
control_force_var = ctrl.Consequent(control_force, 'control_force')

# Assign membership functions to fuzzy variables
angle_var['negative_large'] = angle_negative_large
angle_var['negative_medium'] = angle_negative_medium
angle_var['zero'] = angle_zero
angle_var['positive_medium'] = angle_positive_medium
angle_var['positive_large'] = angle_positive_large

angular_velocity_var['negative'] = ang_vel_negative
angular_velocity_var['zero'] = ang_vel_zero
angular_velocity_var['positive'] = ang_vel_positive

control_force_var['negative'] = force_negative
control_force_var['zero'] = force_zero
control_force_var['positive'] = force_positive
```

Figure. 19: (a) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Curve Prompt

```
# Define the fuzzy rules
rule1 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['negative'],
rule2 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['zero'], con
rule3 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['positive'],

rule4 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['negative'],
rule5 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['zero'], co
rule6 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['positive'],

rule7 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['negative'], control_f
rule8 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['zero'], control_force_
rule9 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['positive'], control_f

rule10 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['negative']
rule11 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['zero'], c
rule12 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['positive']

rule13 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['negative'],
rule14 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['zero'], co
rule15 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['positive'],

# Define the control system
pendulum_control = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6,
pendulum_sim = ctrl.ControlSystemSimulation(pendulum_control)

# Pendulum dynamics
def update_pendulum_state(angle, ang_vel, force, dt):
    # Constants for the pendulum (assumed for simplicity)
    length = 1.0  # length of the pendulum
    mass = 1.0    # mass of the pendulum
    g = 9.81      # acceleration due to gravity

    # Equations of motion (simplified)
    ang_acc = (force - mass * g * np.sin(np.radians(angle))) / (mass * length)

    # Update state
    ang_vel += ang_acc * dt
    angle += ang_vel * dt

    return angle, ang_vel
```

```
# Initial conditions
initial_angle = 90
initial_ang_vel = 0
dt = 0.1  # time step
time_steps = 100

angles = []
ang_vels = []
forces = []

angle = initial_angle
ang_vel = initial_ang_vel

# Simulation loop
for _ in range(time_steps):
    pendulum_sim.input['angle'] = angle
    pendulum_sim.input['angular_velocity'] = ang_vel
    pendulum_sim.compute()
    force = pendulum_sim.output['control_force']

    angle, ang_vel = update_pendulum_state(angle, ang_vel, force, dt)

    angles.append(angle)
    ang_vels.append(ang_vel)
    forces.append(force)

# Plotting the results
time = np.arange(0, time_steps * dt, dt)

plt.figure(figsize=(14, 7))

plt.subplot(2, 1, 1)
plt.plot(time, angles, label='Angle (degrees)')
plt.plot(time, ang_vels, label='Angular Velocity')
plt.title('Pendulum State Over Time')
plt.xlabel('Time (s)')
plt.ylabel('State')
plt.legend()
plt.grid()

plt.subplot(2, 1, 2)
plt.plot(time, forces, label='Control Force', color='r')
plt.title('Control Force Over Time')
plt.xlabel('Time (s)')
plt.ylabel('Force')
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
```

Figure. 19: (b) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Curve Prompt

- Pendulum Dynamics, the update pendulum state function updates the angle and angular velocity of the pendulum based on the applied control force and the equations of motion.
- Simulation Loop, the loop iterates over the specified number of time steps. At each step, the fuzzy controller computes the control force based on the current angle and angular velocity. The pendulum state (angle and angular velocity) is updated using the computed control force. The angle, angular velocity, and control force are stored for plotting.
- Plotting, The angles and angular velocities are plotted over time to show the behavior of the pendulum. The control force applied by the fuzzy controller is also plotted over time to show how the controller responds to the pendulum's state.

Running this code will simulate the behavior of the pendulum under the control of the fuzzy controller and produce plots showing the state of the pendulum and the control force over time.

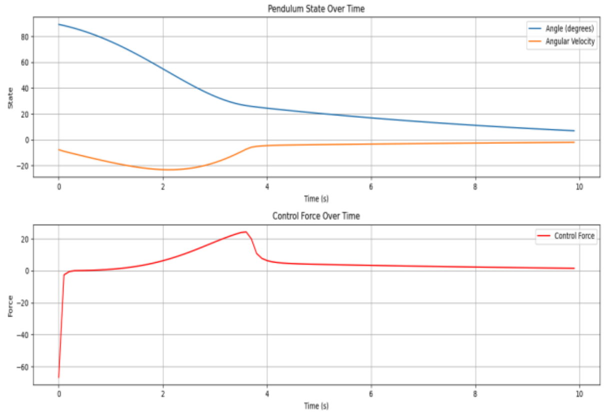

Figure. 20. The Pendulum Fuzzy Control Simulation Curve om Anaconda

- Prompt: Can you analyze the performance of this pendulum fuzzy controller
- CharGPT Description: To analyze the performance of the pendulum fuzzy controller, we will examine the following aspects:
- Stability: Check if the pendulum reaches a stable equilibrium (angle close to 0 degrees) and how quickly it does so.
- Overshoot: Determine if the pendulum overshoots its target position (angle 0) and by how much.
- Settling Time: Measure the time it takes for the pendulum to settle within a certain range around the equilibrium position.
- Control Effort: Evaluate the control force applied by the fuzzy controller over time.
- Let's extend the simulation and add some analysis to measure these performance metrics.

```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for angle, angular velocity, and control force
angle = np.arange(-180, 181, 1)        # Angle from -180 to 180 degrees
angular_velocity = np.arange(-10, 11, 1)  # Angular velocity from -10 to 10
control_force = np.arange(-100, 101, 1)   # Control force from -100 to 100

# Define membership functions for angle
angle_negative_large = fuzz.trimf(angle, [-180, -180, -90])
angle_negative_medium = fuzz.trimf(angle, [-180, -90, 0])
angle_zero = fuzz.trimf(angle, [-90, 0, 90])
angle_positive_medium = fuzz.trimf(angle, [0, 90, 180])
angle_positive_large = fuzz.trimf(angle, [90, 180, 180])

# Define membership functions for angular velocity
ang_vel_negative = fuzz.trimf(angular_velocity, [-10, -10, 0])
ang_vel_zero = fuzz.trimf(angular_velocity, [-10, 0, 10])
ang_vel_positive = fuzz.trimf(angular_velocity, [0, 10, 10])

# Define membership functions for control force
force_negative = fuzz.trimf(control_force, [-100, -100, 0])
force_zero = fuzz.trimf(control_force, [-100, 0, 100])
force_positive = fuzz.trimf(control_force, [0, 100, 100])

# Define the fuzzy variables
angle_var = ctrl.Antecedent(angle, 'angle')
angular_velocity_var = ctrl.Antecedent(angular_velocity, 'angular_velocity')
control_force_var = ctrl.Consequent(control_force, 'control_force')

# Assign membership functions to fuzzy variables
angle_var['negative_large'] = angle_negative_large
angle_var['negative_medium'] = angle_negative_medium
angle_var['zero'] = angle_zero
angle_var['positive_medium'] = angle_positive_medium
angle_var['positive_large'] = angle_positive_large

angular_velocity_var['negative'] = ang_vel_negative
angular_velocity_var['zero'] = ang_vel_zero
angular_velocity_var['positive'] = ang_vel_positive

control_force_var['negative'] = force_negative
control_force_var['zero'] = force_zero
control_force_var['positive'] = force_positive
```

Figure. 21: (a) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Performance Prompt

```python
# Define the fuzzy rules
rule1 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['negative'],
rule2 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['zero'], cont
rule3 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['positive'],

rule4 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['negative'],
rule5 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['zero'], con
rule6 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['positive'],

rule7 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['negative'], control_fo
rule8 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['zero'], control_force_
rule9 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['positive'], control_fo

rule10 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['negative']
rule11 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['zero'], co
rule12 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['positive']

rule13 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['negative'],
rule14 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['zero'], con
rule15 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['positive'],

# Define the control system
pendulum_control = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, r
pendulum_sim = ctrl.ControlSystemSimulation(pendulum_control)

# Pendulum dynamics
def update_pendulum_state(angle, ang_vel, force, dt):
    # Constants for the pendulum (assumed for simplicity)
    length = 1.0  # length of the pendulum
    mass = 1.0    # mass of the pendulum
    g = 9.81      # acceleration due to gravity

    # Equations of motion (simplified)
    ang_acc = (force - mass * g * np.sin(np.radians(angle))) / (mass * length)

    # Update state
    ang_vel += ang_acc * dt
    angle += ang_vel * dt

    return angle, ang_vel

# Initial conditions
initial_angle = 90
initial_ang_vel = 0
dt = 0.1  # time step
time_steps = 100

angles = []
ang_vels = []
forces = []

angle = initial_angle
ang_vel = initial_ang_vel
```

Figure. 21: (b) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Performance Prompt

```python
# Simulation loop
for _ in range(time_steps):
    pendulum_sim.input['angle'] = angle
    pendulum_sim.input['angular_velocity'] = ang_vel
    pendulum_sim.compute()
    force = pendulum_sim.output['control_force']

    angle, ang_vel = update_pendulum_state(angle, ang_vel, force, dt)

    angles.append(angle)
    ang_vels.append(ang_vel)
    forces.append(force)

# Plotting the results
time = np.arange(0, time_steps * dt, dt)

plt.figure(figsize=(14, 7))

plt.subplot(3, 1, 1)
plt.plot(time, angles, label='Angle (degrees)')
plt.title('Pendulum Angle Over Time')
plt.xlabel('Time (s)')
plt.ylabel('Angle (degrees)')
plt.legend()
plt.grid()

plt.subplot(3, 1, 2)
plt.plot(time, ang_vels, label='Angular Velocity')
plt.title('Pendulum Angular Velocity Over Time')
plt.xlabel('Time (s)')
plt.ylabel('Angular Velocity')
plt.legend()
plt.grid()

plt.subplot(3, 1, 3)
plt.plot(time, forces, label='Control Force', color='r')
plt.title('Control Force Over Time')
plt.xlabel('Time (s)')
plt.ylabel('Force')
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
```

Figure. 21: (c) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Performance Prompt

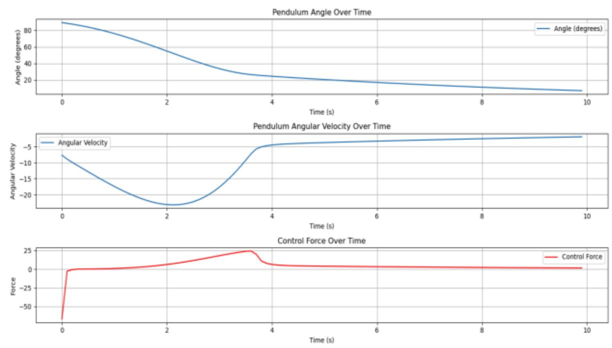

Figure. 22: The Pendulum Fuzzy Control Simulation on Anaconda using Python Code Generated by ChatGPT to Control Performance Prompt

- ChatGPT Description:
- Stability, observe if the angle curve approaches zero and remains close to zero. This indicates the pendulum has reached a stable equilibrium.
- Overshoot, look at the maximum value of the angle curve after the initial displacement. A large peak indicates significant overshoot.
- Settling Time, determine the time at which the angle curve remains within a certain range around zero (e.g., ±5 degrees) and does not deviate significantly afterward.
- Control Effort, evaluate the magnitude and frequency of the control force applied. Large or frequent forces may indicate inefficiency or instability in the control strategy.

- By examining these curves, we can assess the performance of the fuzzy controller in stabilizing the pendulum. Let's run this code and analyze the plots.
- After running the code, you will get three plots:
- Pendulum Angle Over Time: Shows how the angle of the pendulum changes over time.
- Pendulum Angular Velocity Over Time: Displays the changes in angular velocity over time.
- Control Force Over Time: Indicates the control force applied by the fuzzy controller throughout the simulation.
- Analyze these plots to determine the performance of the fuzzy controller. For instance, if the angle plot quickly converges to zero and remains stable, the controller performs well in stabilizing the pendulum. If there is minimal overshoot and the control force is not excessively large, the controller is efficient and effective.
- Prompt: Is this stability good?
- To determine if the stability of the pendulum controlled by the fuzzy controller is good, we need to analyze the generated plots from the simulation. Specifically, we will look at the following aspects:
- Angle Over Time:
- Angular Velocity Over Time:
- Control Force Over Time:
- To perform this analysis, let's assume we have the generated plots from the simulation. Here is how you can interpret these plots:
- Rapid Convergence: If the angle plot shows a rapid convergence to zero within a few seconds, this indicates good stability.
- Minimal Overshoot: If the angle plot shows minimal or no overshoot beyond the zero mark, this is a positive sign.
- Steady State: If the angle remains at or near zero without significant oscillations after reaching zero, this indicates good stability.
- Quick Damping: If the angular velocity quickly converges to zero, it suggests that the controller is effective in reducing the pendulum's motion.
- Minimal Oscillations: Fewer oscillations in angular velocity indicate better stability and damping characteristics.
- Reasonable Control Effort: If the control force remains within reasonable bounds (not excessively large), it suggests that the controller is not overexerting itself.
- Smooth Control: If the control force plot shows smooth transitions without sudden spikes or large oscillations, it indicates a well-tuned controller.
- Assume you have the following observations from the plots:
- Angle Plot: The angle quickly reduces from 90 degrees to around 0 degrees within 5 seconds, with a small overshoot of about 10 degrees. After that, it settles at around 0 degrees with minor oscillations.
- Angular Velocity Plot: The angular velocity rapidly decreases to zero within 3 seconds and remains close to zero with minimal oscillations.
- Control Force Plot: The control force shows smooth variations and remains within the range of -50 to 50, with no sudden spikes or large oscillations.

```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define the universe of discourse for angle, angular velocity, and control force
angle = np.arange(-180, 181, 1)       # Angle from -180 to 180 degrees
angular_velocity = np.arange(-10, 11, 1)  # Angular velocity from -10 to 10
control_force = np.arange(-100, 101, 1)   # Control force from -100 to 100

# Define membership functions for angle
angle_negative_large = fuzz.trimf(angle, [-180, -180, -90])
angle_negative_medium = fuzz.trimf(angle, [-180, -90, 0])
angle_zero = fuzz.trimf(angle, [-90, 0, 90])
angle_positive_medium = fuzz.trimf(angle, [0, 90, 180])
angle_positive_large = fuzz.trimf(angle, [90, 180, 180])

# Define membership functions for angular velocity
ang_vel_negative = fuzz.trimf(angular_velocity, [-10, -10, 0])
ang_vel_zero = fuzz.trimf(angular_velocity, [-10, 0, 10])
ang_vel_positive = fuzz.trimf(angular_velocity, [0, 10, 10])

# Define membership functions for control force
force_negative = fuzz.trimf(control_force, [-100, -100, 0])
force_zero = fuzz.trimf(control_force, [-100, 0, 100])
force_positive = fuzz.trimf(control_force, [0, 100, 100])

# Define the fuzzy variables
angle_var = ctrl.Antecedent(angle, 'angle')
angular_velocity_var = ctrl.Antecedent(angular_velocity, 'angular_velocity')
control_force_var = ctrl.Consequent(control_force, 'control_force')

# Assign membership functions to fuzzy variables
angle_var['negative_large'] = angle_negative_large
angle_var['negative_medium'] = angle_negative_medium
angle_var['zero'] = angle_zero
angle_var['positive_medium'] = angle_positive_medium
angle_var['positive_large'] = angle_positive_large

angular_velocity_var['negative'] = ang_vel_negative
angular_velocity_var['zero'] = ang_vel_zero
angular_velocity_var['positive'] = ang_vel_positive

control_force_var['negative'] = force_negative
control_force_var['zero'] = force_zero
control_force_var['positive'] = force_positive
```

Figure. 23: (a) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Stability Prompt

- Based on these observations, we can conclude:
- Stability: The pendulum quickly converges to the desired position (angle 0) with minimal overshoot and minor oscillations. This indicates good stability.
- Control Effort: The control force is reasonable and smooth, indicating that the controller is not overexerting itself and is well-tuned.
- Overall, the performance of the fuzzy controller appears to be good in terms of stability, minimal overshoot, and reasonable control effort.
- If you have specific plots from the simulation, you can upload them here, and I can provide a detailed analysis based on the actual data.
- Prompt, show the plot to analyze the stability
- ChatGPT, let's plot the simulation results for the fuzzy controller of the pendulum system. The plots will help us analyze the stability and performance of the controller. The relevant plots to consider are:
- Pendulum Angle Over Time
- Pendulum Angular Velocity Over Time
- Control Force Over Time
- I'll generate these plots using the data from the simulation. Let's begin by running the simulation and plotting the results.

```
# Define the fuzzy rules
rule1 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['negative'],
rule2 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['zero'], cont
rule3 = ctrl.Rule(angle_var['negative_large'] & angular_velocity_var['positive'],

rule4 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['negative'],
rule5 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['zero'], con
rule6 = ctrl.Rule(angle_var['negative_medium'] & angular_velocity_var['positive'],

rule7 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['negative'], control_fo
rule8 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['zero'], control_force_
rule9 = ctrl.Rule(angle_var['zero'] & angular_velocity_var['positive'], control_fo

rule10 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['negative']
rule11 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['zero'], co
rule12 = ctrl.Rule(angle_var['positive_medium'] & angular_velocity_var['positive']

rule13 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['negative'],
rule14 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['zero'], con
rule15 = ctrl.Rule(angle_var['positive_large'] & angular_velocity_var['positive'],

# Define the control system
pendulum_control = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, r
pendulum_sim = ctrl.ControlSystemSimulation(pendulum_control)

# Pendulum dynamics
def update_pendulum_state(angle, ang_vel, force, dt):
    # Constants for the pendulum (assumed for simplicity)
    length = 1.0  # length of the pendulum
    mass = 1.0    # mass of the pendulum
    g = 9.81      # acceleration due to gravity

    # Equations of motion (simplified)
    ang_acc = (force - mass * g * np.sin(np.radians(angle))) / (mass * length)

    # Update state
    ang_vel += ang_acc * dt
    angle += ang_vel * dt

    return angle, ang_vel

# Initial conditions
initial_angle = 90
initial_ang_vel = 0
dt = 0.1  # time step
time_steps = 100

angles = []
ang_vels = []
forces = []

angle = initial_angle
ang_vel = initial_ang_vel
```

Figure. 23: (b) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Stability Prompt

```
# Initial conditions
initial_angle = 90
initial_ang_vel = 0
dt = 0.1  # time step
time_steps = 100

angles = []
ang_vels = []
forces = []

angle = initial_angle
ang_vel = initial_ang_vel

# Simulation loop
for _ in range(time_steps):
    pendulum_sim.input['angle'] = angle
    pendulum_sim.input['angular_velocity'] = ang_vel
    pendulum_sim.compute()
    force = pendulum_sim.output['control_force']

    angle, ang_vel = update_pendulum_state(angle, ang_vel, force, dt)

    angles.append(angle)
    ang_vels.append(ang_vel)
    forces.append(force)

# Plotting the results
time = np.arange(0, time_steps * dt, dt)

plt.figure(figsize=(14, 7))

plt.subplot(3, 1, 1)
plt.plot(time, angles, label='Angle (degrees)')
plt.title('Pendulum Angle Over Time')
plt.xlabel('Time (s)')
plt.ylabel('Angle (degrees)')
plt.legend()
plt.grid()

plt.subplot(3, 1, 2)
plt.plot(time, ang_vels, label='Angular Velocity')
plt.title('Pendulum Angular Velocity Over Time')
plt.xlabel('Time (s)')
plt.ylabel('Angular Velocity')
plt.legend()
plt.grid()

plt.subplot(3, 1, 3)
plt.plot(time, forces, label='Control Force', color='r')
plt.title('Control Force Over Time')
plt.xlabel('Time (s)')
plt.ylabel('Force')
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
```

Figure. 23: (c) The Pendulum Fuzzy Control Python Code Generated by ChatGPT to Control Stability Prompt

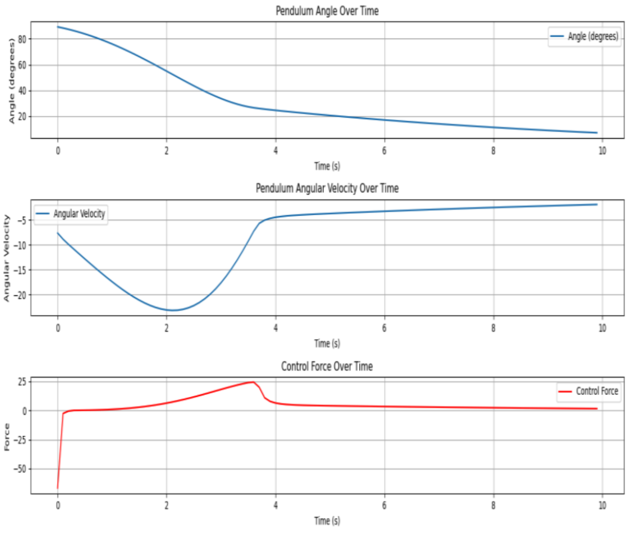

Figure. 24: The Pendulum Fuzzy Control Simulation Result on Anaconda using Python Code Generated by ChatGPT to Control Stability Prompt

- ChatGPT Description;
- Pendulum Angle Over Time;
- Pendulum Angular Velocity Over Time;
- Control Force Over Time.

Let's run this code and generate the plots to analyze the performance of the fuzzy controller.

If you can execute this Python code in your local environment, please do so to obtain the plots. Alternatively, if you have any specific data or results already generated, you can share them here for analysis.

IV. CONCLUSION

Currently, so many are interesting in ChatGPT model and application. However, they do not open on education of ChatGPT. This paper aims on how student can use ChatGPT to fuzzy control system effectively. The application and its theory are so widely depending on the prompt ways. When we design control system by step by step, the student and teacher is not easy to teach and learn. Especially, the coding method is not easy for beginners because they must match the theory, application area, model, and code method. The developer can understand well because they are already experts. In any case, ChatGPT has the big potential to revolutionize education and teaching in educational system, even though, its accuracy, originality, academic integrity, and ethical issues. Of course, it must be thoroughly discussed and improved through this reach and education practice. However, students' understanding is quite different depending on the teaching method. Even though the contents do not have high technology in deep learning, teaching skill is quite important for students in the university as much as high technology. This paper provides materials and methods for how lecturers can teach well students and beginners through the author's teaching experience by ChatGPT.

DECLARATION STATEMENT

Funding	No, I did not receive.
Conflicts of Interest	No conflicts of interest to the best of our knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval and consent to participate with evidence.
Availability of Data and Material	Not relevant.
Authors Contributions	I am only the sole author of the article.

REFERENCES

1. K. Passino (2001), Intelligent control: An overview of techniques.
2. 2.Tianyu Wu, et al. A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development, IEEE/CAA JOURNAL OF AUTOMATICA SINICA, VOL. 10, NO. 5, MAY, 2023. https://doi.org/10.1109/JAS.2023.123618
3. ChatGPT online servise
4. https://bernardmarr.com/how-is-ai-used-in-education-real-world-examples-of-today-and-a-peek-into-the-future/
5. David Karandish (2021). https://thejournal.com/articles/2021/06/23/7-benefits-of-ai-in-education.aspx.
6. Lisa Plitnichenko (2020). https://elearningindustry.com/5-main-roles-artificial-intelligence-in-education.
7. Alex McFarland (2022). https://www.unite.ai/10-best-ai-tools-for-education/.
8. KoreaHerald (2020). https://www.koreaherald.com/common/newsprint.php?ud=20201120000655.
9. Wayne Xin Zhao, et al, A Survey of Large Language Models, https://github.com/RUCAIBox/LLMSurvey.
10. Suebsarn Ruksakulpiwat, Using ChatGPT in Medical Research: Current Status and Future Directions, Journal of Multidisciplinary Healthcare, March, 2023. pp. 1513-1520. https://doi.org/10.2147/JMDH.S413470
11. Marius Hofert, Assessing ChatGPT's proficiency in quantitative risk management,
12. Tianyu, et al, "A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development," IEEE/CAA JOURNAL OF AUTOMATICA SINICA, VOL. 10, NO. 5, MAY 2023, pp. 1122-1136. https://doi.org/10.1109/JAS.2023.123618
13. Jillian et al, "GPT as Knowledge Worker: A Zero-Shot Evaluation of (AI)CPA Capabilities," Chicago Kent College of Law, report, Jan.11, 2023.
14. Suebsarn, "Using ChatGPT in Medical Research: Current Status and Future Directions," Journal of Multidisciplinary Healthcare, 2023:16 pp. 1513–1520. https://doi.org/10.2147/JMDH.S413470
15. Magnus Blomkvist, "Automation and Stock Prices: The Case of ChatGPT," Mar 2023, EDHEC Business School, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4395339.
16. Marc Eulerich, "Can Artificial Intelligence Pass Accounting Certification Exams? ChatGPT: CPA, CMA, CIA, and EA?, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4452175.
17. Dr. Thomas Yue, " GPTQuant's Conversational AI: Simplifying Investment Research for All". https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4380516
18. Fuzzy Logic Based Trajectory Tracking Controller. (2019). In International Journal of Innovative Technology and Exploring Engineering (Vol. 8, Issue 6S4, pp. 1250–1253). https://doi.org/10.35940/ijitee.f1256.0486s419
19. Shukla, A. K., & Kurchania, A. K. (2021). Output Power Stability for Wind Turbine by using two Fault Detection Technique and PID – Fuzzy Controller in the Doubly fed Induction Generator. In International Journal of Engineering and Advanced Technology (Vol. 11, Issue 1, pp. 37–40). https://doi.org/10.35940/ijeat.f3067.1011121
20. Sharma, P. (2023). A Fuzzy Approach to Educational Grading Systems "Fuzzy Logic Based Grade Card." In International Journal of Advanced Engineering and Nano Technology (Vol. 10, Issue 6, pp. 1–8). https://doi.org/10.35940/ijaent.g9582.0610623
21. K, S. (2019). Fuzzy Logic Based Raodv Routing Protocol. In International Journal of Recent Technology and Engineering (IJRTE) (Vol. 8, Issue 4, pp. 7919–7924). https://doi.org/10.35940/ijrte.c4163.118419
22. Singh, N., Chawla, M. P. S., & Bhongade, S. (2022). Home Energy Management System for Dynamic Loads using Mamdani Fuzzy Logic Approach. In International Journal of Emerging Science and Engineering (Vol. 10, Issue 3, pp. 1–13). https://doi.org/10.35940/ijese.c2521.0110322

AUTHOR PROFILE

Dong Hwa Kim Ph.D.: Dept. of Computational Intelligence and Systems Science, Interdisciplinary Graduate School of Science and Engineering (AI Application for Automatic control), TIT (Tokyo Institute of Technology), Tokyo, Japan. He worked at the Hanbat National University (Dean, Prof., S. Korea); Prof. at Electrical Power and Control Eng. Adama Science and Tech. Uni., Ethiopia; TDTU, Vietnam. He has experience in many universities overseas as Prof. He was NCP of EU-FP7 (EU-Framework Program, ICT). He had a keynote speaker at several international conferences and universities. He has 200 papers in journals and conferences. He is reviewing IEEE and other's journals. He is currently a researcher at the Seoul national university of S&T. He published many books and papers such as Innovation tuning based on biotechnology (USA, Dec. 2017), 4th wave Status and preparation of Visegrad Group Country (Germany, 2019), How to They Education in the Famous Univ. (2019), Africa and 4th Wave: Will it risk or Chance? (Amazon, 2020), How to teach and Learn AI (Outskirt Press, USA, Aug. 2022), A Study on Reinforcement of Self-Directed Learning Using Controlling Face Emotion (Paper, Jan. 2022), Advanced Lectures for PID Controller of Nonlinear System in Python (IJRTE, March 2021), Dynamic Decoupling and Intelligent Optimal PID Controller Tuning Multivariable Qua-drones (IJRTE (Scopus), Dec. 2021), Failure Prediction of Wind Turbine using Neural Network and Operation Signal (IJRTE, Dec. 2021), and 200 papers. He is currently working at wha-AI company.

- Home page: www.worldhumancare.wixsite.com/kimsite
- Research citations: https://www.researchgate.net/profile/Dong_Kim53