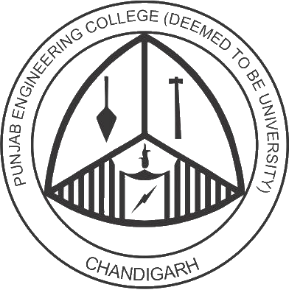
**Project**

*For the course*

**Neural Networks and Fuzzy Logic**



*Submitted to:*

**Rintu Khanna**

Professor & Head

Department of Electrical Engineering

Punjab Engineering College (Deemed to be University) Chandigarh

*Submitted by:*

**Sanket Kumar Singh**

**(SID: 20104137)**

Department of Electrical Engineering

Punjab Engineering College (Deemed to be University) Chandigarh

***VII - SEMESTER (2324-1)***

**DC Motor Speed Prediction Using Fuzzy Logic – Python Implementation**

**Understanding the Problem and Logic Building**

**How Speed and Torque are Defined in an Electric Motor**

**Speed**

The speed of a motor is defined as the rate at which the motor rotates and is measured in revolutions per minute, or RPM. In other words, the speed is defined as how many times the motor spins in 1 minute.

**Torque**

The torque output of a motor is the amount of rotational force that the motor develops and is measured in Newton-meters, or Nm. Simply put, torque is the twisting force of the motor but can be more difficult to understand than its counterpart speed.

**Torque and Speed Relationship: An Analogy**

What is a simple way to understand the torque and speed relationship? Imagine a scenario where an individual has been asked to hammer a long nail into a wall. How many times the individual strikes the nail on the head in one minute is the speed, how hard each contact is with the nail, is the torque. However, the individual has a limited amount of energy, and increasing the force on each strike, or increasing the number of strikes per minute would result in higher energy consumption and would cause the individual to tire out faster.

**What is the Torque and Speed Relationship?**

In an electric motor, the torque and speed relationship is defined by the formula: mechanical power equals the speed multiplied by the torque. The torque and speed relationship is inversely proportional since the rated output power of a motor is a fixed value. As output speed increases, the available output torque decreases proportionately. As the output torque increases, the output speed decreases proportionately. Although speed governs the maximum speed of an electric motor, having more torque enables the system to reach top speed in less time. The dynamic requirements for the intended application will determine whether it is better to have more torque or more speed.

The equation N = kV/T is often used to describe the basic relationship between speed(N), voltage(V), and torque(T) in DC motors.

The relationship between the speed (N), voltage (V), and torque (T) in a DC motor can be described using fuzzy linguistic terms.

Few relations are:

* If the voltage (V) is low and the torque (T) is low, then the speed (N) will likely be slow.
* With a moderate voltage (V) and moderate torque (T), the speed (N) is expected to be balanced and moderate.
* At high voltage (V) and low torque (T), the speed (N) may be fast, but caution is needed as low torque could affect stability.
* When both voltage (V) and torque (T) are high, the speed (N) is likely to be very fast, indicating a powerful motor performance.
* For low voltage (V) and high torque (T), the speed (N) might be limited due to the lower voltage, even with high torque.
* At moderate voltage (V) and high torque (T), the speed (N) should be relatively fast, demonstrating a good balance between voltage and torque.
* If voltage (V) is very high and torque (T) is very low, the speed (N) is expected to be extremely fast, but stability may be a concern.

These relations capture the interactions between voltage, torque, and speed in a fuzzy linguistic manner, considering different combinations of input values.

**Building Membership Functions**

Membership functions in fuzzy logic help define the degree to which a value belongs to a fuzzy set. Three membership functions for voltage (V), torque (T), and speed (N). We'll use linguistic terms like "Very Low", "Low", "Medium", "High" and "Very High" for each variable.

1. Voltage (V) Membership Functions:

* Very Low: V is between 0 and 10 V.
* Low: V is between 4 and 14 V.
* Medium: V is between 8 and 18 V.
* High: V is between 12 and 22 V.
* Very High: V is between 16 and 25 V.

2. Torque (T) Membership Functions:

Since Torque is inversely proportional to the speed, hence in linguistic term, I am calling a “Low” when the torque is high.

* Very Low: T is between 1.8 and 5.0 Nm.
* Low: T is between 1.2 and 4.0 Nm.
* Medium: T is between 0.8 and 3.0 Nm.
* High: T is between 0.4 and 2.0 Nm.
* Very High: T is between 0 and 1.5 Nm.

3. Speed (N) Membership Functions:

* Very Slow: N is between 0 and 400 rpm.
* Slow: N is between 200 and 600 rpm.
* Moderate: N is between 400 and 1000 rpm.
* Fast: N is between 700 and 1200 rpm.
* Very Fast: N is between 1000 and 1500 rpm.

I've used triangular membership functions for the fuzzy sets in the voltage, torque, and speed variables. Triangular membership functions are chosen for their simplicity and effectiveness in many fuzzy systems. They offer a balanced representation of gradual change within a given range.

However, other membership functions like trapezoidal, Gaussian, or more complex shapes could be used based on specific system requirements.

**Python Implementation**

To implement fuzzy logic in Python, we can use a library called scikit-fuzzy. Scikit-fuzzy is a Python library that provides tools and methods for fuzzy logic systems.

Let’s first install the library:

pip install scikit-fuzzy

Now, let’s use this library and implement the logic:

**CODE**

import numpy as np

import skfuzzy as fuzz

from skfuzzy import control as ctrl

import streamlit as st

import matplotlib.pyplot as plt

import matplotlib

matplotlib.rc('legend', fontsize=6)

# Fuzzy logic setup

voltage = ctrl.Antecedent(np.arange(0, 25, 1), 'voltage')

torque = ctrl.Antecedent(np.arange(0, 15, 1), 'torque')

speed = ctrl.Consequent(np.arange(0, 1500, 1), 'speed')

# Membership functions

voltage['Very Low'] = fuzz.trimf(voltage.universe, [0, 5, 10])

voltage['Low'] = fuzz.trimf(voltage.universe, [4, 9, 14])

voltage['Medium'] = fuzz.trimf(voltage.universe, [8, 13, 18])

voltage['High'] = fuzz.trimf(voltage.universe, [12, 17, 22])

voltage['Very High'] = fuzz.trimf(voltage.universe, [16, 21, 25])

torque['Very Low'] = fuzz.trimf(torque.universe, [18, 25, 25])

torque['Low'] = fuzz.trimf(torque.universe, [12, 20, 25])

torque['Medium'] = fuzz.trimf(torque.universe, [8, 15, 20])

torque['High'] = fuzz.trimf(torque.universe, [4, 10, 15])

torque['Very High'] = fuzz.trimf(torque.universe, [0, 5, 10])

speed['Very Slow'] = fuzz.trimf(speed.universe, [0, 200, 400])

speed['Slow'] = fuzz.trimf(speed.universe, [200, 500, 600])

speed['Moderate'] = fuzz.trimf(speed.universe, [400, 600, 1000])

speed['Fast'] = fuzz.trimf(speed.universe, [700, 900, 1200])

speed['Very Fast'] = fuzz.trimf(speed.universe, [1000, 1300, 1500])

# Fuzzy rules (5X5 = 25)

rule1 = ctrl.Rule(voltage['Very Low'] & torque['Very Low'], speed['Very Slow'])

rule2 = ctrl.Rule(voltage['Low'] & torque['Very Low'], speed['Very Slow'])

rule3 = ctrl.Rule(voltage['Medium'] & torque['Very Low'], speed['Very Slow'])

rule4 = ctrl.Rule(voltage['High'] & torque['Very Low'], speed['Slow'])

rule5 = ctrl.Rule(voltage['Very High'] & torque['Very Low'], speed['Slow'])

rule6 = ctrl.Rule(voltage['Very Low'] & torque['Low'], speed['Very Slow'])

rule7 = ctrl.Rule(voltage['Low'] & torque['Low'], speed['Slow'])

rule8 = ctrl.Rule(voltage['Medium'] & torque['Low'], speed['Slow'])

rule9 = ctrl.Rule(voltage['High'] & torque['Low'], speed['Moderate'])

rule10 = ctrl.Rule(voltage['Very High'] & torque['Low'], speed['Moderate'])

rule11 = ctrl.Rule(voltage['Very Low'] & torque['Medium'], speed['Slow'])

rule12 = ctrl.Rule(voltage['Low'] & torque['Medium'], speed['Moderate'])

rule13 = ctrl.Rule(voltage['Medium'] & torque['Medium'], speed['Moderate'])

rule14 = ctrl.Rule(voltage['High'] & torque['Medium'], speed['Fast'])

rule15 = ctrl.Rule(voltage['Very High'] & torque['Medium'], speed['Fast'])

rule16 = ctrl.Rule(voltage['Very Low'] & torque['High'], speed['Moderate'])

rule17 = ctrl.Rule(voltage['Low'] & torque['High'], speed['Fast'])

rule18 = ctrl.Rule(voltage['Medium'] & torque['High'], speed['Fast'])

rule19 = ctrl.Rule(voltage['High'] & torque['High'], speed['Very Fast'])

rule20 = ctrl.Rule(voltage['Very High'] & torque['High'], speed['Very Fast'])

rule21 = ctrl.Rule(voltage['Very Low'] & torque['Very High'], speed['Fast'])

rule22 = ctrl.Rule(voltage['Low'] & torque['Very High'], speed['Very Fast'])

rule23 = ctrl.Rule(voltage['Medium'] & torque['Very High'], speed['Very Fast'])

rule24 = ctrl.Rule(voltage['High'] & torque['Very High'], speed['Very Fast'])

rule25 = ctrl.Rule(voltage['Very High'] & torque['Very High'], speed['Very Fast'])

# control system, and simulator definitions

system = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9, rule10, rule11, rule12, rule13, rule14, rule15, rule16, rule17, rule18, rule19, rule20, rule21, rule22, rule23, rule24, rule25])

simulator = ctrl.ControlSystemSimulation(system)

# Create a Streamlit app

st.set\_page\_config(page\_title='DC Motor Speed Predictor', page\_icon= 'images/favicon.png')

st.title('Fuzzy Logic Speed Control System')

# Input values (simulating user input)

voltage\_input = st.slider('Voltage Input', min\_value=0.0, max\_value=25.0, value=15.0)

torque\_input = st.slider('Torque Input', min\_value=0.0, max\_value=25.0, value=15.0)

if voltage\_input < 0 or voltage\_input > 25 or torque\_input < 0 or torque\_input > 15:

    st.warning("Input values should be between 0 and 25.")

else:

    simulator.input['voltage'] = voltage\_input

    simulator.input['torque'] = torque\_input

    try:

        simulator.compute()

        result = round(simulator.output['speed'], 2)

        st.markdown(f"Calculated DC Motor Speed Output: \*\*{result}\*\*")

        st.markdown("---")

        plot\_width, plot\_height = 6, 3

        # Generate the membership functions plot for voltage

        voltage\_membership\_fig, ax = plt.subplots(figsize=(plot\_width, plot\_height))

        ax.plot(voltage.universe, fuzz.trimf(voltage.universe, [0, 5, 10]), label='Very Low')

        ax.plot(voltage.universe, fuzz.trimf(voltage.universe, [4, 9, 14]), label='Low')

        ax.plot(voltage.universe, fuzz.trimf(voltage.universe, [8, 13, 18]), label='Medium')

        ax.plot(voltage.universe, fuzz.trimf(voltage.universe, [12, 17, 22]), label='High')

        ax.plot(voltage.universe, fuzz.trimf(voltage.universe, [16, 21, 25]), label='Very High')

        ax.fill\_between(voltage.universe, fuzz.interp\_membership(voltage.universe, fuzz.trimf(voltage.universe, [0, 5, 10]), voltage\_input), alpha=0.2)

        ax.fill\_between(voltage.universe, fuzz.interp\_membership(voltage.universe, fuzz.trimf(voltage.universe, [4, 9, 14]),  voltage\_input), alpha=0.2)

        ax.fill\_between(voltage.universe, fuzz.interp\_membership(voltage.universe, fuzz.trimf(voltage.universe, [8, 13, 18]), voltage\_input), alpha=0.2)

        ax.fill\_between(voltage.universe, fuzz.interp\_membership(voltage.universe, fuzz.trimf(voltage.universe, [12, 17, 22]), voltage\_input), alpha=0.2)

        ax.fill\_between(voltage.universe, fuzz.interp\_membership(voltage.universe, fuzz.trimf(voltage.universe, [16, 21, 25]), voltage\_input), alpha=0.2)

        ax.legend()

        ax.set\_xlabel('Voltage')

        ax.set\_ylabel('Membership')

        st.write("Voltage Membership Function:")

        st.pyplot(voltage\_membership\_fig)

        # Generate the membership functions plot for torque

        torque\_membership\_fig, ax = plt.subplots(figsize=(plot\_width, plot\_height))

        ax.plot(torque.universe, fuzz.trimf(torque.universe, [18, 25, 25]), label='Very Low')

        ax.plot(torque.universe, fuzz.trimf(torque.universe, [12, 20, 25]), label='Low')

        ax.plot(torque.universe, fuzz.trimf(torque.universe, [8, 15, 20]), label='Medium')

        ax.plot(torque.universe, fuzz.trimf(torque.universe, [4, 10, 15]), label='High')

        ax.plot(torque.universe, fuzz.trimf(torque.universe, [0, 5, 10]), label='Very High')

        ax.fill\_between(torque.universe, fuzz.interp\_membership(torque.universe, fuzz.trimf(torque.universe, [18, 25, 25]), torque\_input), alpha=0.2)

        ax.fill\_between(torque.universe, fuzz.interp\_membership(torque.universe, fuzz.trimf(torque.universe, [12, 20, 25]), torque\_input), alpha=0.2)

        ax.fill\_between(torque.universe, fuzz.interp\_membership(torque.universe, fuzz.trimf(torque.universe, [8, 15, 20]), torque\_input), alpha=0.2)

        ax.fill\_between(torque.universe, fuzz.interp\_membership(torque.universe, fuzz.trimf(torque.universe, [4, 10, 15]), torque\_input), alpha=0.2)

        ax.fill\_between(torque.universe, fuzz.interp\_membership(torque.universe, fuzz.trimf(torque.universe, [0, 5, 10]), torque\_input), alpha=0.2)

        ax.legend()

        ax.set\_xlabel('Torque')

        ax.set\_ylabel('Membership')

        st.write("Torque Membership Function:")

        st.pyplot(torque\_membership\_fig)

        st.markdown("\*The changing colors in the membership function plots represent the degree of membership of the input value in each fuzzy set.\*")

        st.markdown("---")

        # Generate the membership functions plot for speed

        speed\_membership\_fig, ax = plt.subplots(figsize=(plot\_width, plot\_height))

        ax.plot(speed.universe, fuzz.trimf(speed.universe, [0, 200, 400]), label='Very Slow')

        ax.plot(speed.universe, fuzz.trimf(speed.universe, [200, 500, 600]), label='Slow')

        ax.plot(speed.universe, fuzz.trimf(speed.universe, [400, 600, 1000]), label='Moderate')

        ax.plot(speed.universe, fuzz.trimf(speed.universe, [700, 900, 1200]), label='Fast')

        ax.plot(speed.universe, fuzz.trimf(speed.universe, [1000, 1300, 1500]), label='Very Fast')

        ax.fill\_between(speed.universe, fuzz.interp\_membership(speed.universe, fuzz.trimf(speed.universe, [0, 200, 400]), result), alpha=0.2)

        ax.fill\_between(speed.universe, fuzz.interp\_membership(speed.universe, fuzz.trimf(speed.universe, [200, 500, 600]), result), alpha=0.2)

        ax.fill\_between(speed.universe, fuzz.interp\_membership(speed.universe, fuzz.trimf(speed.universe, [400, 600, 1000]), result), alpha=0.2)

        ax.fill\_between(speed.universe, fuzz.interp\_membership(speed.universe, fuzz.trimf(speed.universe, [700, 900, 1200]), result), alpha=0.2)

        ax.fill\_between(speed.universe, fuzz.interp\_membership(speed.universe, fuzz.trimf(speed.universe, [1000, 1300, 1500]), result), alpha=0.2)

        ax.legend()

        ax.set\_xlabel('Speed')

        ax.set\_ylabel('Membership')

        st.write("Speed Membership Function:")

        st.pyplot(speed\_membership\_fig)

        st.markdown("\*The colors in the speed membership function plot represent the degree of membership of the output value in each fuzzy set.\*")

        st.markdown("\*So, for example, if an output value has a high degree of membership in the 'Moderate' set, we see a larger colored area under the 'Moderate' curve.\*")

    except ValueError as e:

        st.warning("Error: Crisp output cannot be calculated. Please check the input values and rules.")

st.markdown("---")

st.markdown(

    """

    <div style=" bottom: 10px; width: 100%; text-align: center;">

        <p style="color: #888;">

            <a href="https://github.com/imsanketsingh" style="color: #888;">Github</a> |

            <a href="https://www.linkedin.com/in/imsanketsingh/" style="color: #888;">Linkedin</a> |

            <a href="https://github.com/imsanketsingh/DC-Motor-Speed-Control-Using-Fuzzy-Logic/" style="color: #888;">Code</a>

        </p>

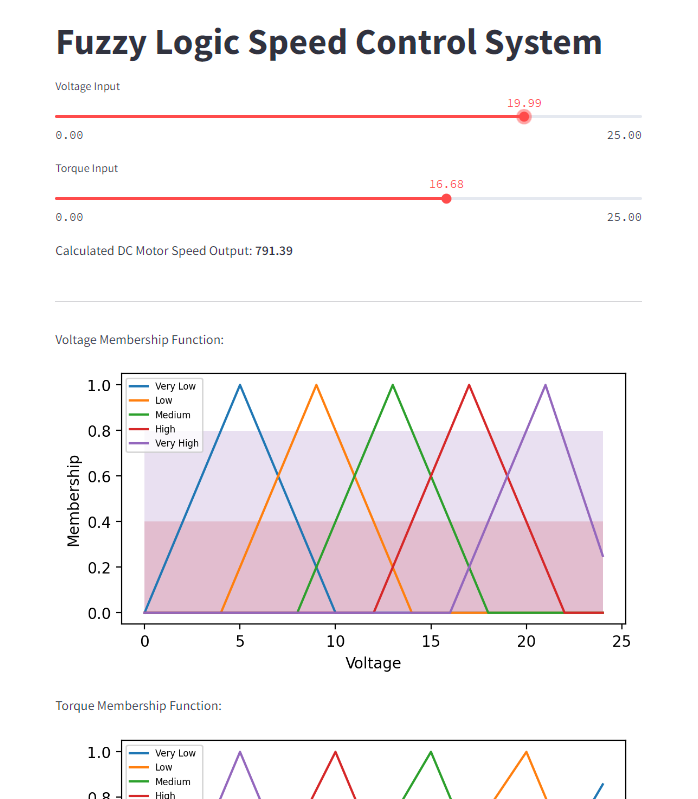
    </div>

    """,

    unsafe\_allow\_html=True

)

**OUTPUT**

****

I have created a small web application that does the same (taking voltage and torque & predicting the speed, with dynamic membership plots) and it can be accessed from anywhere using this link: [**https://fuzzyspeedcontrol.streamlit.app/**](https://fuzzyspeedcontrol.streamlit.app/)

The future plan is to implement various problems using the different fuzzy logics using a lot of different membership functions.