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

**Understanding the Problem and Logic Building**

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 5 V.
* Low: V is between 4 and 10 V.
* Medium: V is between 8 and 15 V.
* High: V is between 12 and 20 V.
* Very High: V is above 18 V.

2. Torque (T) Membership Functions:

* Very Low: T is between 0 and 5 Nm.
* Low: T is between 4 and 10 Nm.
* Medium: T is between 8 and 15 Nm.
* High: T is between 12 and 20 Nm.
* Very High: T is above 18 Nm.

3. Speed (N) Membership Functions:

* Very Slow: N is between 0 and 500 rpm.
* Slow: N is between 400 and 1000 rpm.
* Moderate: N is between 800 and 1500 rpm.
* Fast: N is between 1200 and 2000 rpm.
* Very Fast: N is above 1800 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

# Fuzzy variables for voltage, torque, and speed are created. These variables define the universe of discourse for each variable.

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

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

speed = ctrl.Consequent(np.arange(0, 2500, 1), 'speed')

# 3 membership functions

# Membership functions are defined for each linguistic term of the variables using trimf method with is used to create the triangular membership curves

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

voltage['Low'] = fuzz.trimf(voltage.universe, [4, 10, 15])

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

voltage['High'] = fuzz.trimf(voltage.universe, [12, 18, 25])

voltage['Very High'] = fuzz.trimf(voltage.universe, [18, 25, 25])

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

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

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

torque['High'] = fuzz.trimf(torque.universe, [12, 18, 25])

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

speed['Very Slow'] = fuzz.trimf(speed.universe, [0, 500, 1000])

speed['Slow'] = fuzz.trimf(speed.universe, [400, 1000, 1500])

speed['Moderate'] = fuzz.trimf(speed.universe, [800, 1500, 2000])

speed['Fast'] = fuzz.trimf(speed.universe, [1200, 2000, 2500])

speed['Very Fast'] = fuzz.trimf(speed.universe, [1800, 2500, 2500])

# Fuzzy rules are established to map combinations of input variables (voltage and torque) to the output variable (speed). Each rule is structured as an 'if-then' statement using the defined linguistic terms.

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

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

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

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

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

# A control system is created by combining all the defined rules. Then, a simulation object is generated using this control system.

system = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5])

simulator = ctrl.ControlSystemSimulation(system)

# Input values for voltage and torque are set. The simulation is performed, and the output (Predicted Motor Speed) is computed based on the given input values following the defined fuzzy logic rules.

voltage\_input = float(input("Enter the Voltage(V): "))

torque\_input = float(input("Enter the Torque(Nm): "))

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

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

else:

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

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

    try:

        simulator.compute()

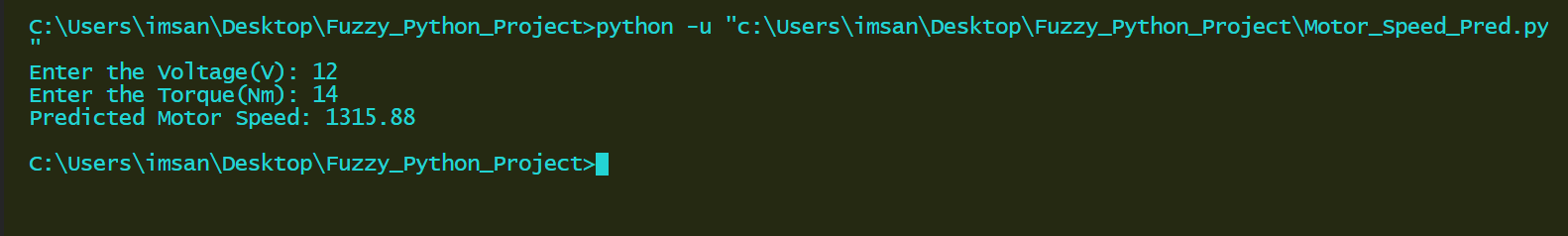
        result = round(simulator.output['speed'], 2)

        print(f"Predicted Motor Speed: {result}")

    except ValueError as e:

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

**OUTPUT**

****

**Further Work**

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.