Fuzzy Logic Power Optimisation and RPM Optimisation

Henri Setyo Pambudi

1 Introduction

This document provides a detailed explanation of a fuzzy logic-based system designed to suggest the optimal RPM (Revolutions Per Minute) and corresponding power output for a given roll angle. The system utilizes fuzzy control and fuzzy inference rules to predict the RPM and power output based on historical data.

2 Dependencies

The following Python libraries are required to execute the code:

- `pandas` – For data manipulation and reading CSV files.
- `numpy` – For numerical computations.
- `skfuzzy` – For fuzzy logic control and inference.
- `matplotlib` – For plotting the results.

3 Code Explanation

3.1 Data Loading and Merging

The code begins by loading two CSV files containing the raw data for RPM and power output, as well as roll angle data. The files are merged based on the common column `Timestamp`:

```
file1 = "/content/drive/MyDrive/Colab Notebooks/
    Dummy_RPM_Power_Output_Data.csv"
file2 = "/content/drive/MyDrive/Colab Notebooks/
    Dummy_Roll_Angle_Data.csv"

data_rpm_power = pd.read_csv(file1)
data_roll_angle = pd.read_csv(file2)

merged_data = pd.merge(data_rpm_power, data_roll_angle, on='
    Timestamp')
```

3.2 Fuzzy Logic Control System Setup

The fuzzy variables (Antecedents and Consequents) are defined as follows:

- `roll_angle` is defined between -30 to 30 degrees. - `rpm` is defined between 0 to 7000 RPM. - `power_output` is defined between 0 to 500 Watts.

The fuzzy sets for the variables are:

- `roll_angle`: low, medium, high
- `rpm`: low, medium, high
- `power_output`: low, medium, high

Membership functions for each fuzzy set are defined using triangular membership functions (`fuzz.trimf`):

```
roll_angle['low'] = fuzz.trimf(roll_angle.universe, [-30, -30, 0])
roll_angle['medium'] = fuzz.trimf(roll_angle.universe, [-30, 0,
    30])
roll_angle['high'] = fuzz.trimf(roll_angle.universe, [0, 30, 30])

rpm['low'] = fuzz.trimf(rpm.universe, [0, 0, 3500])
rpm['medium'] = fuzz.trimf(rpm.universe, [0, 3500, 7000])
rpm['high'] = fuzz.trimf(rpm.universe, [3500, 7000, 7000])

power_output['low'] = fuzz.trimf(power_output.universe, [0, 0,
    250])
power_output['medium'] = fuzz.trimf(power_output.universe, [0, 250,
     500])
power_output['high'] = fuzz.trimf(power_output.universe, [250, 500,
     500])
```

3.3 Fuzzy Inference Rules

The fuzzy inference system is based on 9 rules that map combinations of `roll_angle` and `rpm` to `power_output`. Examples of the rules are:

```
rule1 = ctrl.Rule(roll_angle['low'] & rpm['low'], power_output['low
    '])
rule2 = ctrl.Rule(roll_angle['low'] & rpm['medium'], power_output['
    medium'])
rule3 = ctrl.Rule(roll_angle['low'] & rpm['high'], power_output['
    high'])
```

3.4 Fuzzy Control System Setup

The fuzzy control system is created using the `ControlSystem` and `ControlSystemSimulation` classes from the `skfuzzy` library:

```
power_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5,
     rule6, rule7, rule8, rule9])
power_simulation = ctrl.ControlSystemSimulation(power_ctrl)
```

3.5 RPM Suggestion Function

The function `suggest_rpm` is defined to calculate the optimal RPM for a given roll angle by iterating through possible RPM values and computing the corresponding power output. The RPM that yields the maximum power output is selected as the suggested RPM.

```
def suggest_rpm(roll_angle_value):
    best_rpm = None
    best_power = -float('inf')

    for rpm_value in np.linspace(0, 7000, 100):
        power_simulation.input['roll_angle'] = roll_angle_value
        power_simulation.input['rpm'] = rpm_value
        power_simulation.compute()

        output_power = power_simulation.output['power_output']

        if output_power > best_power:
            best_power = output_power
            best_rpm = rpm_value

    return best_rpm, best_power
```

3.6 Generating RPM Suggestions for the Dataset

The function `get_rpm_suggestions` iterates through the merged dataset, applying the `suggest_rpm` function to generate the suggested RPM and power output for each timestamp.

```
def get_rpm_suggestions(merged_data):
    suggestions = []

    for index, row in merged_data.iterrows():
        roll_angle_value = row['Roll Angle (degrees)']
        suggested_rpm, suggested_power = suggest_rpm(
            roll_angle_value)
        suggestions.append({
            'Timestamp': row['Timestamp'],
            'Roll Angle (degrees)': roll_angle_value,
            'Suggested RPM': suggested_rpm,
            'Suggested Power Output (Watts)': suggested_power
        })

    return pd.DataFrame(suggestions)
```

3.7 Saving the Suggested Data to CSV

The final suggested data, including roll angle, suggested RPM, and suggested power output, is saved to a CSV file.

```
suggested_data.to_csv("/content/drive/MyDrive/Colab Notebooks/
    suggested_rpm_power.csv", index=False)
```

3.8 Plotting the Results

Various plots are created using `matplotlib` to visualize the relationships between the roll angle, RPM, and power output. Scatter plots are used to represent the data:

```
# Roll Angle vs RPM (Raw Data)
plt.scatter(merged_data['Roll Angle (degrees)'], merged_data['RPM'], label='RPM (Raw Data)', color='g', marker='o')
plt.title('Roll Angle vs RPM (Raw Data)')
plt.xlabel('Roll Angle (degrees)')
plt.ylabel('RPM')
plt.grid(True)
plt.legend()
plt.show()

# Roll Angle vs Power Output (Raw Data)
plt.scatter(merged_data['Roll Angle (degrees)'], merged_data['Power Output (Watts)'], label='Power Output (Raw Data)', color='orange', marker='o')
plt.title('Roll Angle vs Power Output (Raw Data)')
plt.xlabel('Roll Angle (degrees)')
plt.ylabel('Power Output (Watts)')
plt.grid(True)
plt.legend()
plt.show()

# Roll Angle vs Suggested RPM
plt.scatter(suggested_data['Roll Angle (degrees)'], suggested_data['Suggested RPM'], label='Suggested RPM', color='b', marker='o')
plt.title('Roll Angle vs Suggested RPM')
plt.xlabel('Roll Angle (degrees)')
plt.ylabel('Suggested RPM')
plt.grid(True)
plt.legend()
plt.show()

# Roll Angle vs Suggested Power Output
plt.scatter(suggested_data['Roll Angle (degrees)'], suggested_data['Suggested Power Output (Watts)'], label='Suggested Power Output', color='r', marker='o')
plt.title('Roll Angle vs Suggested Power Output')
plt.xlabel('Roll Angle (degrees)')
plt.ylabel('Suggested Power Output (Watts)')
plt.grid(True)
plt.legend()
plt.show()

# Suggested Power Output vs Suggested RPM
plt.scatter(suggested_data['Suggested RPM'], suggested_data['Suggested Power Output (Watts)'], label='Suggested Power vs RPM', color='purple', marker='o')
plt.title('Suggested Power Output vs RPM')
plt.xlabel('Suggested RPM')
plt.ylabel('Suggested Power Output (Watts)')
plt.grid(True)
plt.legend()
```

```
44 plt.show()
```

4 Usage

Fuzzy logic here used to optimized power output and RPM. The inputs are roll angle of vehicle, power output, and RPM before optimized using fuzzy logic. But, all of the data here is just dummy data. The csv files contain 600 data each variables.