EPS System Implementation with Fuzzy Logic

Introduction

Electric power steering (EPS) systems have replaced old hydraulic systems as a key component in modern vehicle engineering. EPS improves vehicle handling and driver comfort by adjusting steering assistance based on speed and angle. Integrating fuzzy logic into EPS systems improves performance and safety by providing adaptive and intelligent control options.

ESP system Overview

1.Component

a.Steering Wheel Sensor: Determines the angle and rate of steering input from the driver.
b.The vehicle speed sensor monitors the vehicle's speed.
c.The Electronic Control Unit (ECU) processes sensor data to calculate the degree of steering assistance required.
d.The motor provides the necessary torque to help the driver's steering effort.

2.Operation

a. At low speeds, the technology provides additional assistance, making it easier to navigate.
b. At high speeds, the system decreases assistance to improve stability and control.

Features

1. Adaptive Steering Assistance
2. Improved Driver Feel and Comfort
3. Compensation for Road Conditions
4. Enhancing Safety
5. Real-Time Adaptation

Installation (Google Collaboratory)

In order to run this project from google Collaboratory, follow these steps:

(If jupyter notebook is pre-installed on your pc then you can go through it.)

1**.Clone the Repository**

-open any folder in vscode studio

-open terminal any run the command to clone repository:

git clone https://github.com/emergingravi/EPS-Electric-power-Steering-with-Fuzzy-implimentation

2.**Installation**

- open your google drive

- right click

- select more - connect more apps

- search for google Collaboratory

- install

3.**creating a new file**

- right click

- select more - google Collaboratory

4. **Connecting to Server**

- select dropdown menu of connect

- change runtime

- select your desired hardware accelerator (TPU v2 -recommended)

- save

- click on connect

6**. Run the ESP Algorithm**

-follow the instruction within the Collaboratory to run the ESP_algorithm and view results.

7. **Shutdown Collaboratory**

- once finished, you can close the file by pressing CTRL+W.

Algorithm Details

Input and output variables

Here Antecedent is used for the input for speed and steering angle whereas Consequent is used for output for torque assistance

```
speed = ctrl.Antecedent(np.arange(0, 101, 1), 'speed')
steering_angle = ctrl.Antecedent(np.arange(-45, 46, 1), 'steering_angle')
torque_assistance = ctrl.Consequent(np.arange(0, 101, 1), 'torque_assistance')
```

Membership function

Membership function for speed

```
speed['low'] = fuzz.trimf(speed.universe, [0, 0, 50])
speed['medium'] = fuzz.trimf(speed.universe, [0, 50, 100])
speed['high'] = fuzz.trimf(speed.universe, [50, 100, 100])
```

The speed of the vehicle is typically categorized into three fuzzy sets: ‘low’,’medium’ and ‘high’. These sets are defined using triangular membership functions.

Low speed: Starts at 0 km/h with full membership (1). linearly decreases to 0 membership at 50 km/h.

Medium speed: Starts at 0 km/h with membership (0). Increases to full membership (1) at 50km/h and decreases back to 0 membership at 100 km/h

High Speed: Starts at 50 km/h with no membership. At 100 km/h, membership level increases to full (1). Maintains full membership for speeds greater than 100 km/h.

Membership function for steering angle

```
steering_angle['negative_large'] = fuzz.trimf(steering_angle.universe, [-45, -45, -15])
steering_angle['negative_small'] = fuzz.trimf(steering_angle.universe, [-30, -15, 0])
steering_angle['zero'] = fuzz.trimf(steering_angle.universe, [-5, 0, 5])
steering_angle['positive_small'] = fuzz.trimf(steering_angle.universe, [0, 15, 30])
steering_angle['positive_large'] = fuzz.trimf(steering_angle.universe, [15, 45, 45])
```

The steering angle is also categorized into five fuzzy sets:

Negative Large:

Full membership at -45 degrees.
At -15 degrees, membership decreases to zero.

Negative small

begins at -30 degrees and has no members.
At -15 degrees, increases to full membership.
At zero degrees, membership drops back to 0. begins at -30 degrees and has no members.
At -15 degrees, increases to full membership.
At zero degrees, membership drops back to 0.

Zero

begins at -5 degrees and has no members.
At zero degrees, increases to full membership.
At five degrees, membership drops back to zero.

Positive small

begins with 0 degrees and 0 members.
At fifteen degrees, increases to full membership.
At thirty degrees, membership drops back to zero.

Positive large

begins at 15 degrees and has no members.
At 45 degrees, increases to full membership.
stays at full membership when the angle is more than 45 degrees.

Membership function for torque assistance

Three fuzzy sets comprise the torque assistance:

```
torque_assistance['low'] = fuzz.trimf(torque_assistance.universe, [0, 0, 50])
torque_assistance['medium'] = fuzz.trimf(torque_assistance.universe, [0, 50, 100])
torque_assistance['high'] = fuzz.trimf(torque_assistance.universe, [50, 100, 100])
```

Low torque assistance

Complete enrollment at 0%.
At 50%, membership drops to 0.

Medium torque assistance

Starts at 0% with no membership.
Increases to full membership by 50%.
At 100%, membership decreases back to zero.

High torque assistance

Starts at 50% with no membership.
Increases full membership to 100%.
For values greater than 100%, the membership level remains full.

Fuzzy rules

A set of if-then rules that govern the interaction of input variables and output. The rules are made in such a way that it covers all the possibility.
Like this: "If speed is low and steering angle is large, then torque assistance is high."

```
rules = [
    ctrl.Rule(speed['low'] & steering_angle['negative_large'], torque_assistance['high']
    ctrl.Rule(speed['low'] & steering_angle['negative_small'], torque_assistance['high']
    ctrl.Rule(speed['low'] & steering_angle['zero'], torque_assistance['medium']),
    ctrl.Rule(speed['low'] & steering_angle['positive_small'], torque_assistance['high']
    ctrl.Rule(speed['low'] & steering_angle['positive_large'], torque_assistance['high']

    ctrl.Rule(speed['medium'] & steering_angle['negative_large'], torque_assistance['med
    ctrl.Rule(speed['medium'] & steering_angle['negative_small'], torque_assistance['med
    ctrl.Rule(speed['medium'] & steering_angle['zero'], torque_assistance['low']),
    ctrl.Rule(speed['medium'] & steering_angle['positive_small'], torque_assistance['med
    ctrl.Rule(speed['medium'] & steering_angle['positive_large'], torque_assistance['med

    ctrl.Rule(speed['high'] & steering_angle['negative_large'], torque_assistance['low']
    ctrl.Rule(speed['high'] & steering_angle['negative_small'], torque_assistance['low']
    ctrl.Rule(speed['high'] & steering_angle['zero'], torque_assistance['low']),
    ctrl.Rule(speed['high'] & steering_angle['positive_small'], torque_assistance['low']
    ctrl.Rule(speed['high'] & steering_angle['positive_large'], torque_assistance['low']
]
```

Creating a Control System

- The object (ControlSystem) is defined using the fuzzy rules
- The torque_control is the resulting controlsystem object which encapsulates all the fuzzy logic rules and can be used for simulations
- The object ControlSystemSimulation is created based on ControlSystem
- Torque_simulation is the resulting ControlSystemSimulation object used to run simulation

```
torque_control = ctrl.ControlSystem(rules)
torque_simulation = ctrl.ControlSystemSimulation(torque_control)
```

Input values

The information about the speed and steering angle is taken from the user through the input variables input_speed and input_steering_angle and sent for simulation

```
input_speed =float(input("input the speed (0 to 100)  : ") ) # Example speed in km/h
input_steering_angle =float(input("enter the sterring angle (-45 to 45 ) : "))
torque_simulation.input['speed'] = input_speed
torque_simulation.input['steering_angle'] = input_steering_angle
```

Results

After running the algorithm, the torque assistance is calculated and displayed.

```
torque_simulation.compute()
print(f"Torque assistance: {torque_simulation.output['torque_assistance']}%")
torque_assistance.view(sim=torque_simulation
```

Output

```
input the speed (0 to 100)  : 56
enter the sterring angle (-45 to 45 ) : 20
Torque assistance: 49.61432397579605%
```

GitHub Repository Link:

https://github.com/emergingravi/EPS-Electric-power-Steering-with-Fuzzy-implimentation