LOVELY PROFESSIONAL UNIVERSITY

REPORT FOR FUZZY LOGIC CONTROL OF WIND ENERGY SYSTEM

As a project work for Course

SOFT COMPUTING (INT246)

Name : Markanda Meher

Registration Number : 11904889

Name : Shlok Kumar

Registration Number : 11911460

Name : Nishchay Mishra

Registration Number : 11904865

Program : CSE B.Tech

Semester : 5th

School : School of Computer Science and Engineering

Name of the University : Lovely Professional University

Date of submission : 5th Dec 2021

Lovely Professional University

Jalandhar, Punjab, India

ABSTRACT

Wind energy has gained an increasing worldwide interest due to the continuous increase in fuel cost and the need to have a clean source of energy. The main objective of most of the wind energy systems is to extract the maximum power available in the wind stream. However, the wind regime varies continuously and thus the system controllers should be updated to follow these variations. This paper is intended to apply fuzzy logic control techniques to overcome the effect of the wind speed variations on the parameters of the wind turbines and their controllers.

ACKNOWLEDGMENTS

First of all, we thank The Almighty God for blessing us and supporting us. We take this opportunity to express our profound and sincere gratitude to our mentor Mr. Ishan Kumar for her exemplary guidance, monitoring and constant encouragement throughout this course. We would like to thank our family for the support throughout our study, and for checking over our project report. Lastly, we thank all our friends without whom this project would not be possible

TABLE OF CONTENT

TEAM MEMBERS:-

Markanda Meher :-

Contributions:-

1. Coding(joined)
2. Report(joined)

Shlok Kumar :-

Contributions:-

1. Coding(joined)
2. Report(joined)

Nishchay Mishra :-

Contributions:-

1.Coding(joined)

2.Report(joined)

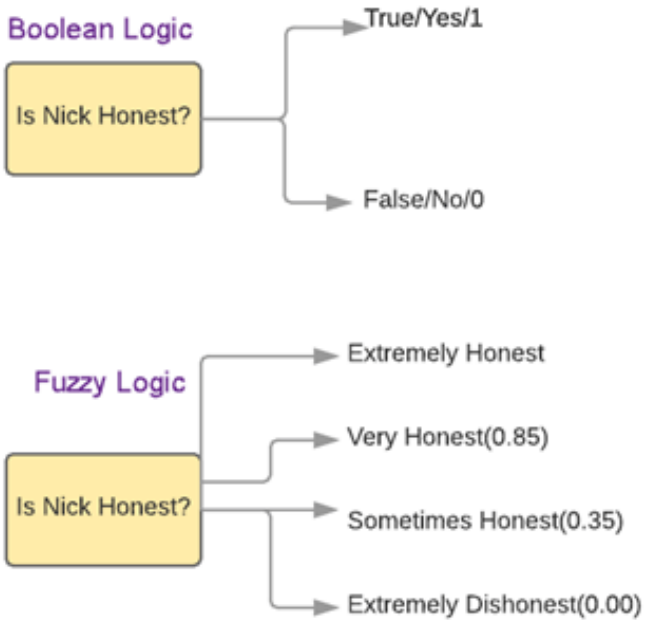
Introduction

What is fuzzy logic??

The **fuzzy** word means the things that are not clear or are vague. Sometimes, we cannot decide in real life that the given problem or statement is either true or false. At that time, this concept provides many values between the true and false and gives the flexibility to find the best solution to that problem.

EXAMPLE

See the below-given diagram. It shows that in a Fuzzy system, the values are denoted by a 0 to 1 number. In this example, 1.0 means absolute truth and 0.0 means absolute falseness.



Fuzzy Logic with Example

LIBRARIES:-

NumPy:-

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

As the whole project is based on whole complex stats ,we will use these fast calculations and provide results.

Pandas:-

Pandas is a fast, powerful, flexible, and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language. We will provide highly optimized performance with back-end source code with the use of Pandas.

Matplotlib:-

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

Opendatasets:-

opendatasets is a Python library for downloading datasets from online sources like Kaggle and Google Drive using a simple Python command.

Scikit-learn:-

It is a Python library is associated with NumPy and SciPy. It is considered as one of the best libraries for working with complex data. There are a lot of changes being made in this library. We will use it for cross-validation feature, providing the ability to use more than one metric. Lots of training methods like logistics regression will be used to provide some little improvements

Scikit-Fuzzy:-

Scikit-Fuzzy is **a collection of fuzzy logic algorithms intended for use in the SciPy Stack**, written in the Python computing language. This SciKit is developed by the SciPy community.

Skfuzzy.control:-

skfuzzy. control subpackage, **providing a high-level API for fuzzy system design**. ... Consequent (output/control) variable for a fuzzy control system. skfuzzy.control.ControlSystem ([rules]) Base class to contain a Fuzzy Control System.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

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

```
import opendatasets as od
```

READING DATASET

```
turbine_df = load_turbine_data()
```

```
turbine_df.tail()
```

	Unnamed: 0	ActivePower	AmbientTemperatue	BearingShaftTemperature	Blade1PitchAngle	Blade2PitchAngle	Blade3PitchAngle	ControlBoxTemperature	Gear
2020-03-30 23:10:00+00:00		70.044465	27.523741	45.711129	1.515669	1.950088	1.950088	0.0	
2020-03-30 23:20:00+00:00		40.833474	27.602882	45.598573	1.702809	2.136732	2.136732	0.0	
2020-03-30 23:30:00+00:00		20.777790	27.560925	45.462045	1.706214	2.139664	2.139664	0.0	
2020-03-30 23:40:00+00:00		62.091039	27.810472	45.343827	1.575352	2.009781	2.009781	0.0	
2020-03-30 23:50:00+00:00		68.664425	27.915828	45.231610	1.499323	1.933124	1.933124	0.0	

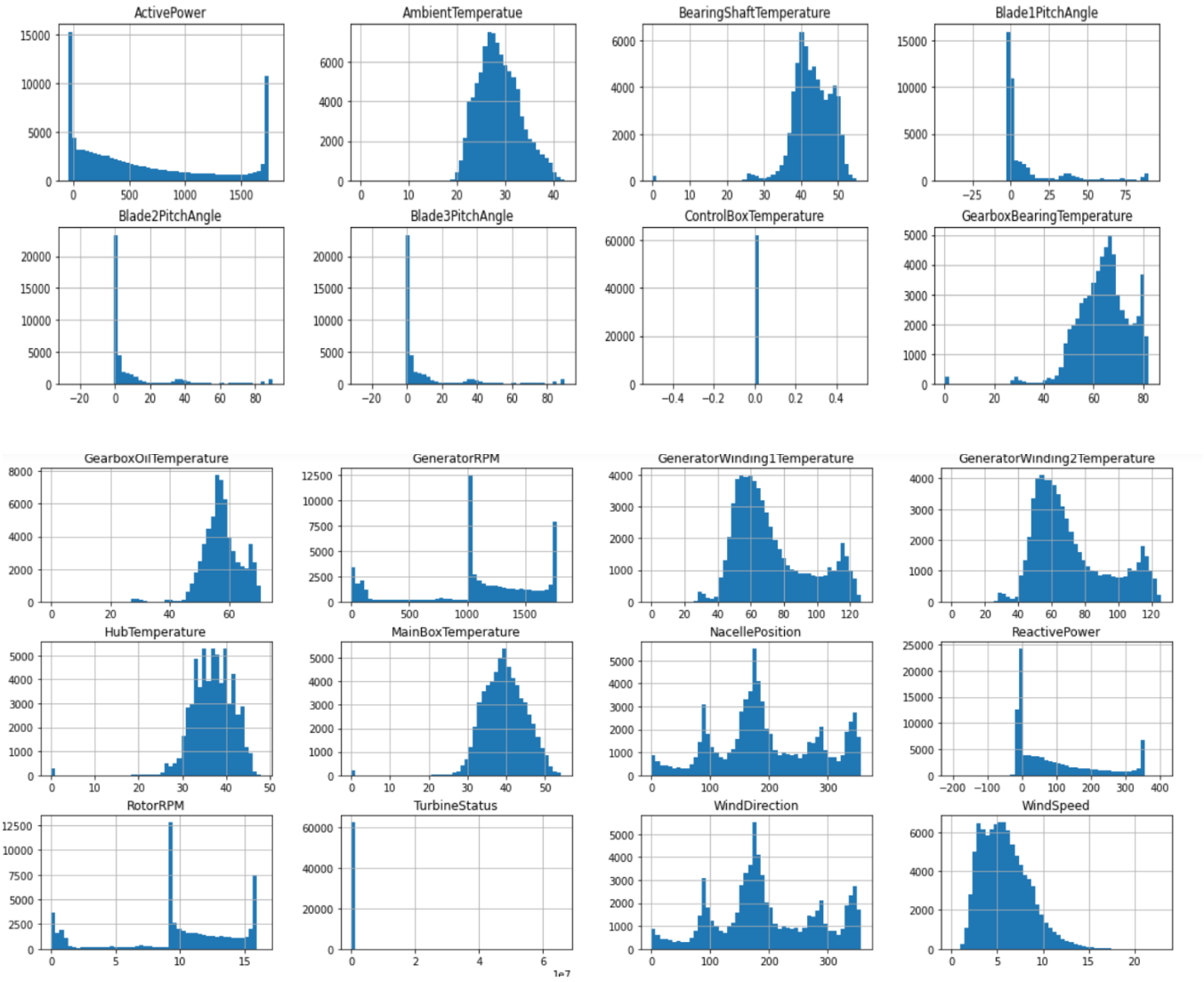
× 22 columns

```
turbine_df.describe()
```

	ActivePower	AmbientTemperatue	BearingShaftTemperature	Blade1PitchAngle	Blade2PitchAngle	Blade3PitchAngle	ControlBoxTemperature	GearboxBear
count	94750.000000	93817.000000	62518.000000	41996.000000	41891.000000	41891.000000	62160.0	
mean	619.109805	28.774654	43.010189	9.749641	10.036535	10.036535	0.0	
std	611.275373	4.369145	5.545312	20.644828	20.270465	20.270465	0.0	
min	-38.524659	0.000000	0.000000	-43.156734	-26.443415	-26.443415	0.0	
25%	79.642258	25.627428	39.840247	-0.939849	-0.433264	-0.433264	0.0	
50%	402.654893	28.340541	42.910877	0.394399	0.888977	0.888977	0.0	
75%	1074.591780	31.664772	47.007976	8.099302	8.480194	8.480194	0.0	
max	1779.032433	42.405597	55.088655	90.143610	90.017830	90.017830	0.0	

visualization

```
turbine_df.hist(bins=50, figsize=(20,15))
plt.show()
```



Training Dataset

The train_test_split function is for splitting a single dataset for two different purposes: training and testing. The training subset is for building your model. The testing subset is for using the model on unknown data to evaluate the performance of the model.

```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(turbine_df, test_size=0.02, random_state=42)
print(f"Rows in train set: {len(train_set)}\nRows in test set: {len(test_set)}")
```

```
Rows in train set: 115859
Rows in test set: 2365
```

```
turbine_df = train_set.drop("ActivePower", axis=1)
turbine_labels = train_set["ActivePower"].copy()
```

Selecting a desired model

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
# model = LinearRegression()
# model = DecisionTreeRegressor()
model = RandomForestRegressor()
model.fit(turbine_df, turbine_labels)
```

```
RandomForestRegressor()
```

```
some_data = turbine_df.iloc[:5]
some_labels = turbine_labels.iloc[:5]
```

S

Evaluating the model

```
from sklearn.metrics import mean_squared_error
predictions = model.predict(turbine_df)
mse = mean_squared_error(turbine_labels, predictions)
rmse = np.sqrt(mse)
```

```
rmse
```

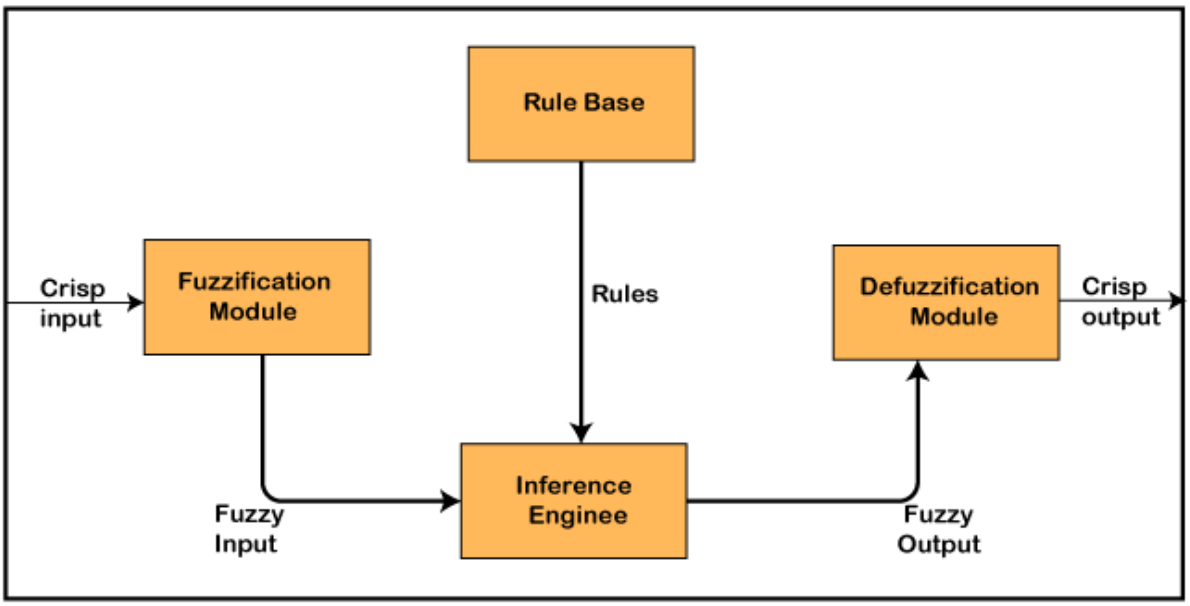
```
11.644682184807994
```

Fuzzy logic controller

In the architecture of the **Fuzzy Logic** system, each component plays an important role. The architecture consists of the different four components which are given below.

1. Rule Base
2. Fuzzification
3. Inference Engine
4. Defuzzification

Following diagram shows the architecture or process of a Fuzzy Logic system:



1. Rule Base

Rule Base is a component used for storing the set of rules and the If-Then conditions given by the experts are used for controlling the decision-making systems. There are so many updates that come in the Fuzzy theory recently, which offers effective methods for designing and tuning of fuzzy controllers. These updates or developments decreases the number of fuzzy set of rules.

2. Fuzzification

Fuzzification is a module or component for transforming the system inputs, i.e., it converts the crisp number into fuzzy steps. The crisp numbers are those inputs which are measured by the sensors and then fuzzification passed them into the control systems for further processing. This component divides the input signals into following five states in any Fuzzy Logic system:

- Large Positive (LP)

- Medium Positive (MP)
- Small (S)
- Medium Negative (MN)
- Large negative (LN)

3. Inference Engine

This component is a main component in any Fuzzy Logic controller (FLC), because all the information is processed in the Inference Engine. It allows users to find the matching degree between the current fuzzy input and the rules. After the matching degree, this system determines which rule is to be added according to the given input field. When all rules are fired, then they are combined for developing the control actions.

4. Defuzzification

Defuzzification is a module or component, which takes the fuzzy set inputs generated by the **Inference Engine**, and then transforms them into a crisp value. It is the last step in the process of a fuzzy logic system. The crisp value is a type of value which is acceptable by the user. Various techniques are present to do this, but the user has to select the best one for reducing the errors.

MEMBERSHIP FUNCTION

Membership functions characterize fuzziness (i.e., all the information in fuzzy set), whether the elements in fuzzy sets are discrete or continuous. Membership functions can be defined as **a technique to solve practical problems by experience rather than knowledge**.

```
# Definig membership function for the change in rotational speed of generator

speed_change=ctrl.Antecedent(np.arange(-100,101),'speed-change')
speed_change['NB']=fuzzy.trimf(speed_change.universe,[-100,-55,-40])
speed_change['NS']=fuzzy.trapmf(speed_change.universe,[-45,-40,-12,-10])
speed_change['Z']=fuzzy.trimf(speed_change.universe,[-12,0,12])
speed_change['PS']=fuzzy.trapmf(speed_change.universe,[10,12,45,50])
speed_change['PB']=fuzzy.trapmf(speed_change.universe,[40,45,95,100])
```

```
# Definig membership function for the active power
active_power=ctrl.Antecedent(model.predict(some_data),'active-power')
active_power['NB']=fuzzy.trimf(active_power.universe,[-38,-19,-14])
active_power['NS']=fuzzy.trimf(active_power.universe,[-12,-8,-4])
active_power['Z']=fuzzy.trimf(active_power.universe,[-4,0,40])
active_power['PS']=fuzzy.trimf(active_power.universe,[300,500,800])
active_power['PB']=fuzzy.trapmf(active_power.universe,[1000,1500,1780])
```

```
disired_change_speed=ctrl.Consequent(np.arange(0,21),'disired_change_speed')
names = ['NB', 'NS', 'Z', 'PS', 'PB']
disired_change_speed.automf(5,names=names)
disired_change_speed.view()
```

```
# Defing the rules
rule1=ctrl.Rule(antecedent = ((speed_change['NB'] & active_power['NB']) |
                (speed_change['Z'] & active_power['PB']) |
                (speed_change['PB'] & active_power['PB'])),
                consequent = disired_change_speed['PB'])


rule2=ctrl.Rule(antecedent = ((speed_change['NB'] & active_power['NS']) |
                (speed_change['Z'] & active_power['PS']) |
                (speed_change['NS'] & active_power['NB']) |
                (speed_change['NS'] & active_power['NS']) |
                (speed_change['PS'] & active_power['PS']) |
                (speed_change['PS'] & active_power['PB']) |
                (speed_change['PB'] & active_power['PS']))
                ,consequent = disired_change_speed['PS'])


rule3=ctrl.Rule(antecedent = ((speed_change['NB'] & active_power['Z']) |
                (speed_change['NS'] & active_power['Z']) |
                (speed_change['Z'] & active_power['Z']) |
                (speed_change['PS'] & active_power['Z']) |
                (speed_change['PB'] & active_power['Z']))
                ,consequent = disired_change_speed['Z'])


rule4=ctrl.Rule(antecedent = ((speed_change['NB'] & active_power['PS'])|
                speed_change['NS'] & active_power['PB'] |
                speed_change['NS'] & active_power['PS'] |
                (speed_change['Z'] & active_power['NS']) |
                (speed_change['Z'] & active_power['NB']) |
                (speed_change['PB'] & active_power['NS']) |
                (speed_change['PS'] & active_power['NS']) |
                (speed_change['PS'] & active_power['NB']))
                , consequent = disired_change_speed['NS'])
```

```
rule5=ctrl.Rule(antecedent = (speed_change['NB'] & active_power['PB']|
                (speed_change['PB'] & active_power['NB']))
                ,consequent = disired_change_speed['NB'])
```

```
rule=[rule1,rule2,rule3,rule4,rule5]
system=ctrl.ControlSystem(rule)
y=ctrl.ControlSystemSimulation(system)

speed_change.view()
power_change.view()
disired_change_speed.view(sim=y)
plt.show()
```

CONCLUSION

- Fuzzy Logic Controller(FLC) is great way to control any machine without much human intervention. Machine can only understand the binary code, but the real-life data is full of vagueness, so FLC is great way to deal this problem. When accurate reasoning is not available, it provides an accurate level of reasoning. Loss of energy and depreciation of machine can be prevented through this technology. FLC can also be used in other way and can also support “one sun ,one grid” initiative more efficiently.

REFERENCE

Dataset:

https://www.kaggle.com/theforcecoder/wind-power-forecasting

GitHub Link:

https://github.com/Markanda100/SoftComputingProject