Automatic Generation of Simulated Data in Dymola for Training of the Deep Learning Model Used in Power System

Shunyao Xu

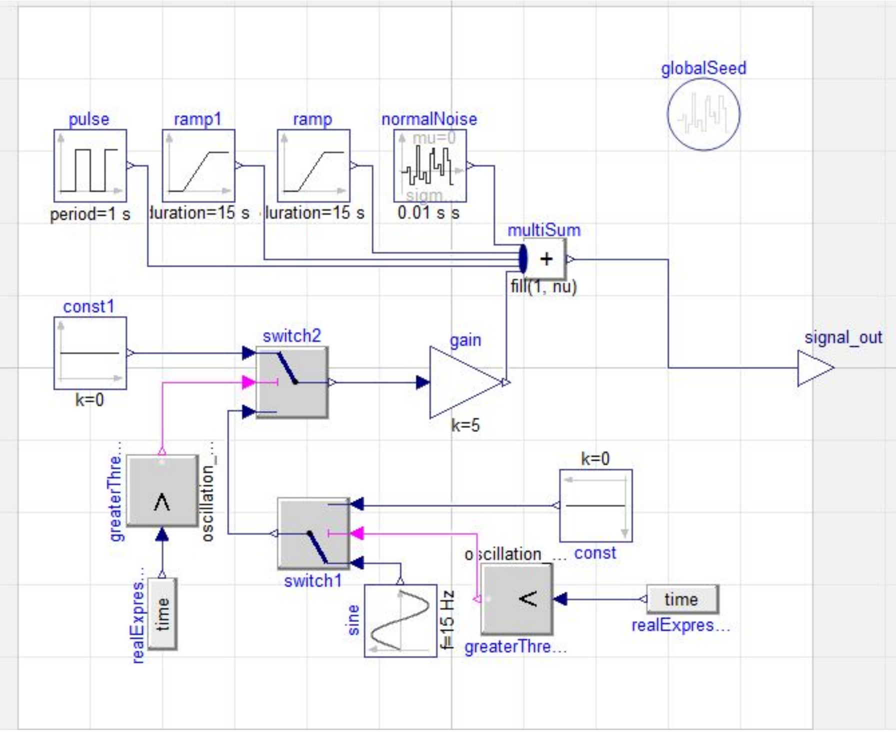
ECSE-4170 Modeling and Simulation for Cyber-Physical Systems
Spring 2021

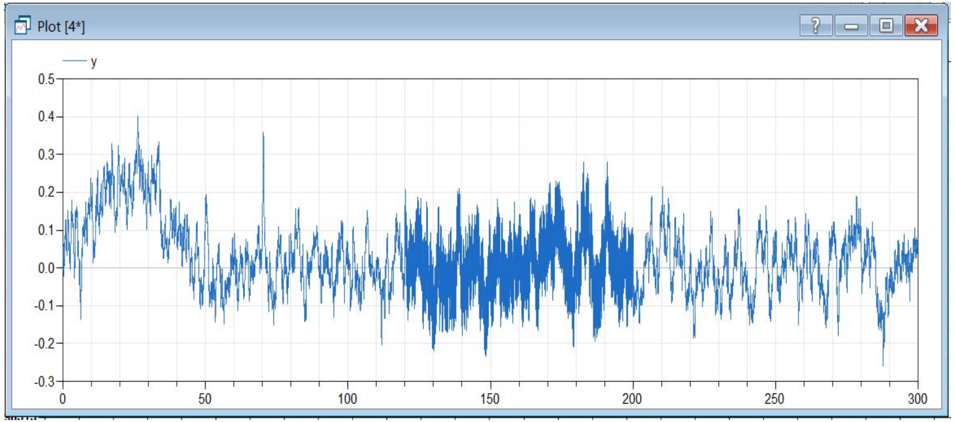
Overview

- Build the Model in Dymola
    - The Signal Model and The Power System Model
    - Construct the Replaceable Model
    - Propagate Parameters
    - Simulate the Model in Dymola
- Data Generation and Model Training in Python
    - Simulate the Model and Extract the Data using Dymola Python Interfaces
    - Training of the ML Model Using the Generated Data
- Automatic Data Generation
- Compare with Previous Models
- Conclusion

Build the Model in Dymola

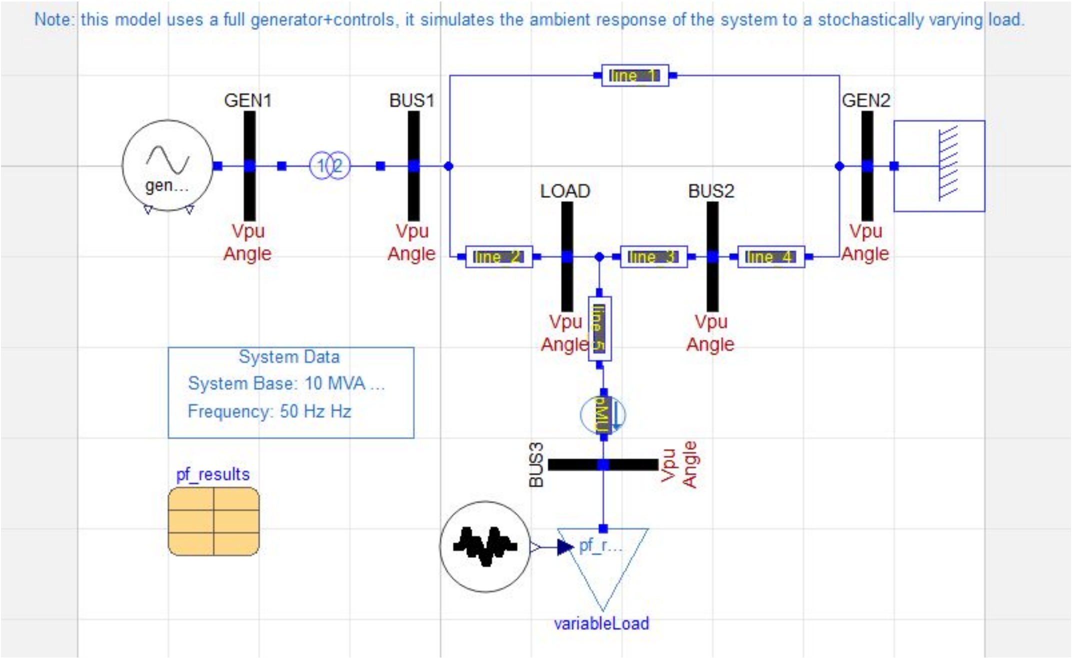
Signal Model

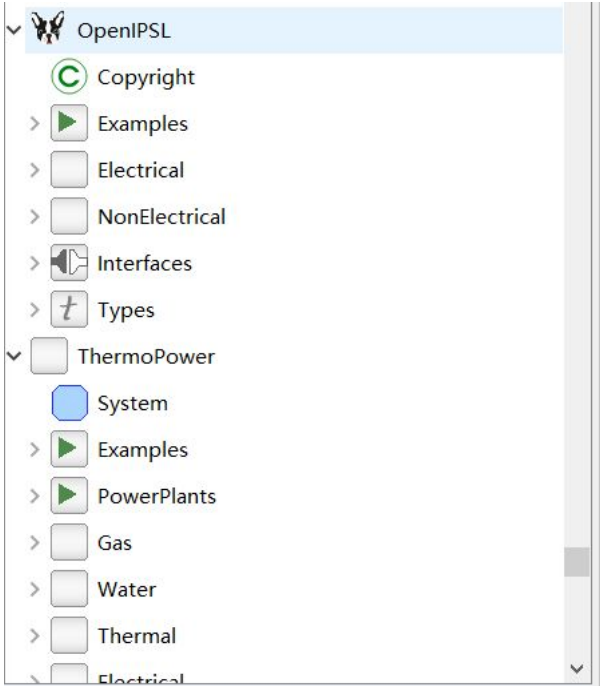




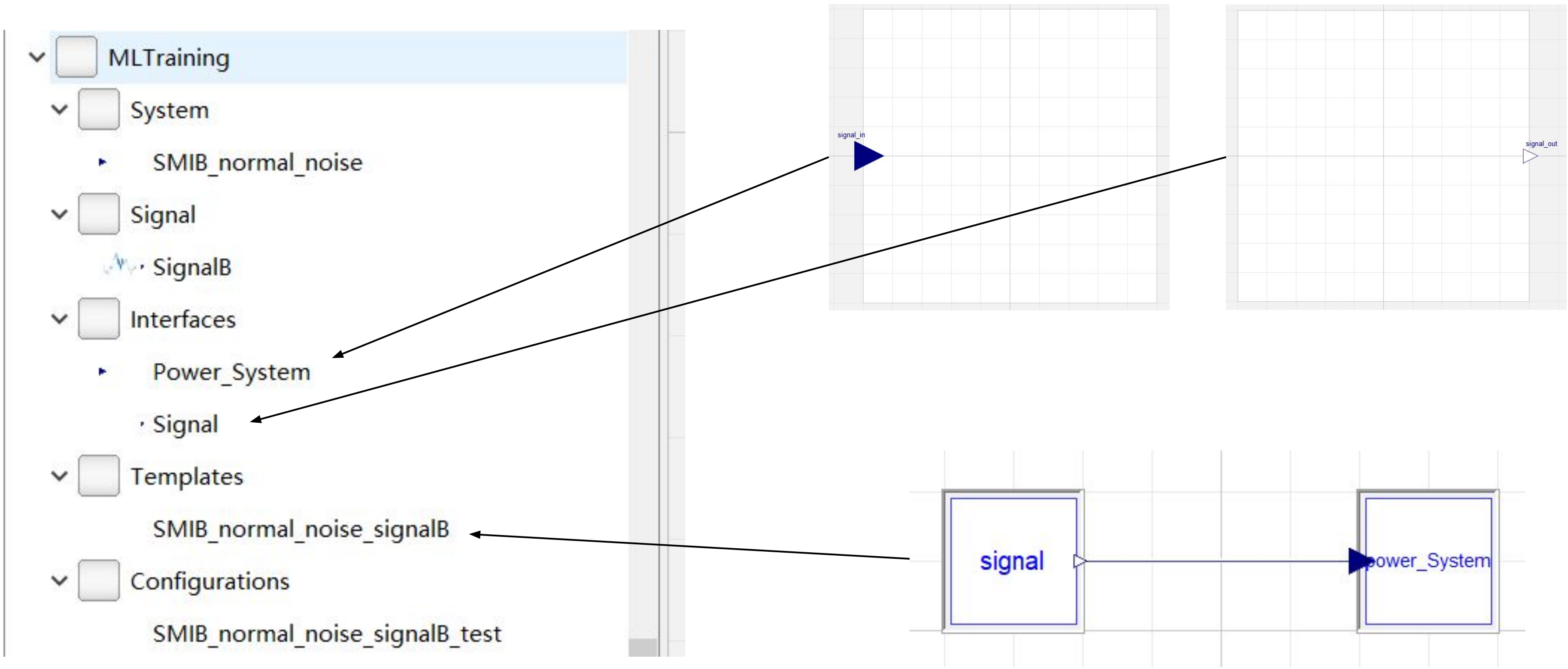
- Generate synthetic data with oscillation
- Amplitude, start time, and end time for the oscillation event is adjustable

Power System Model

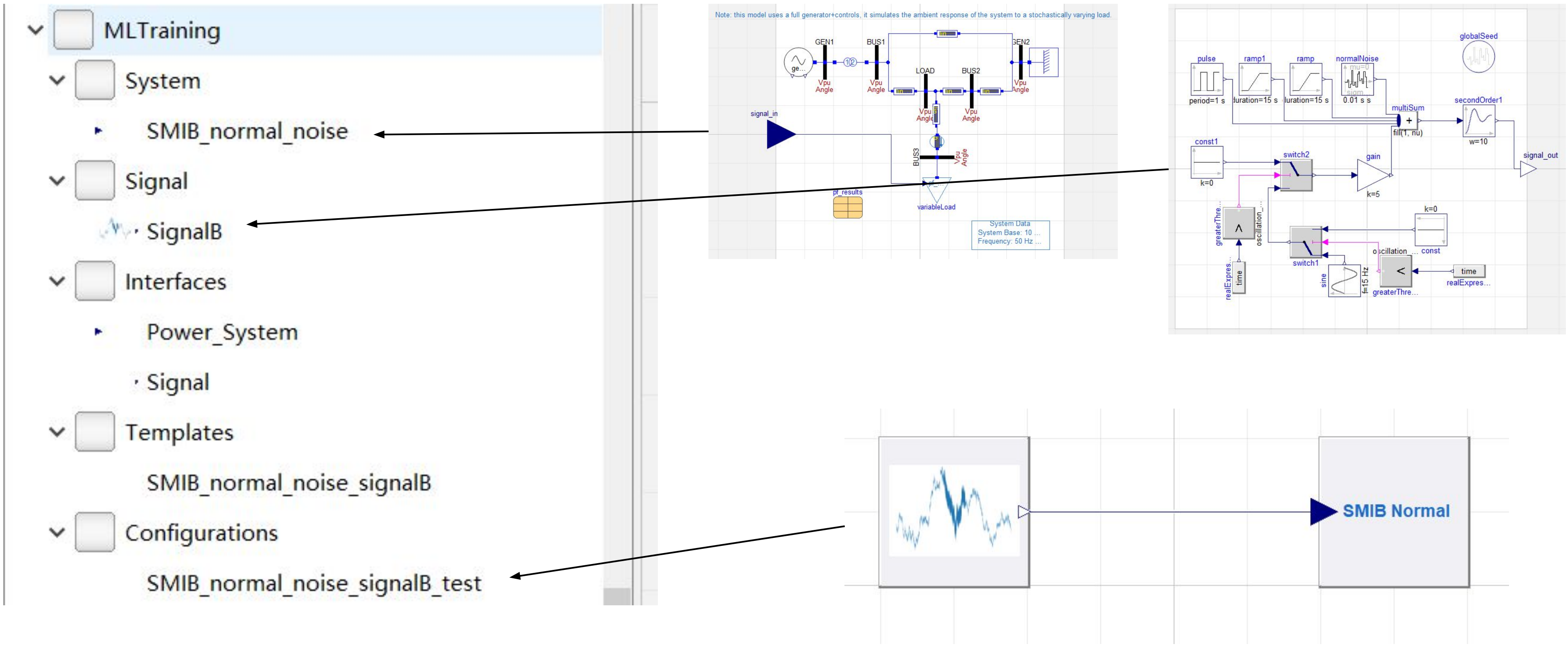




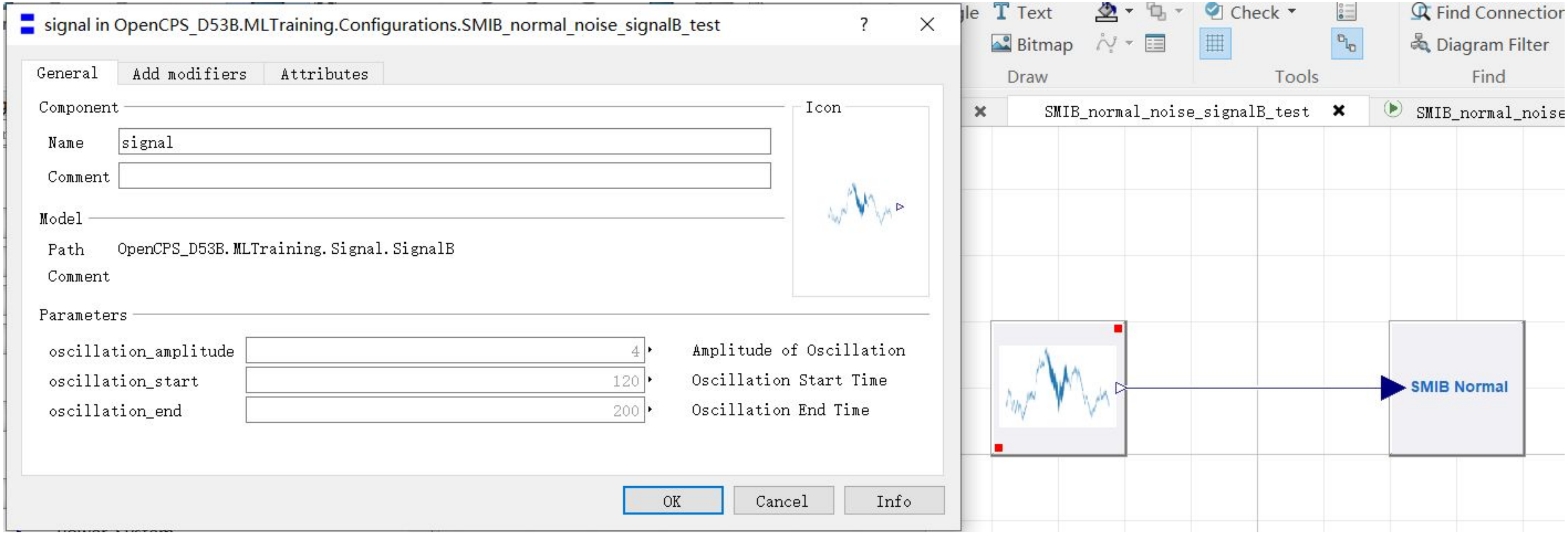
Construct the Replaceable Model



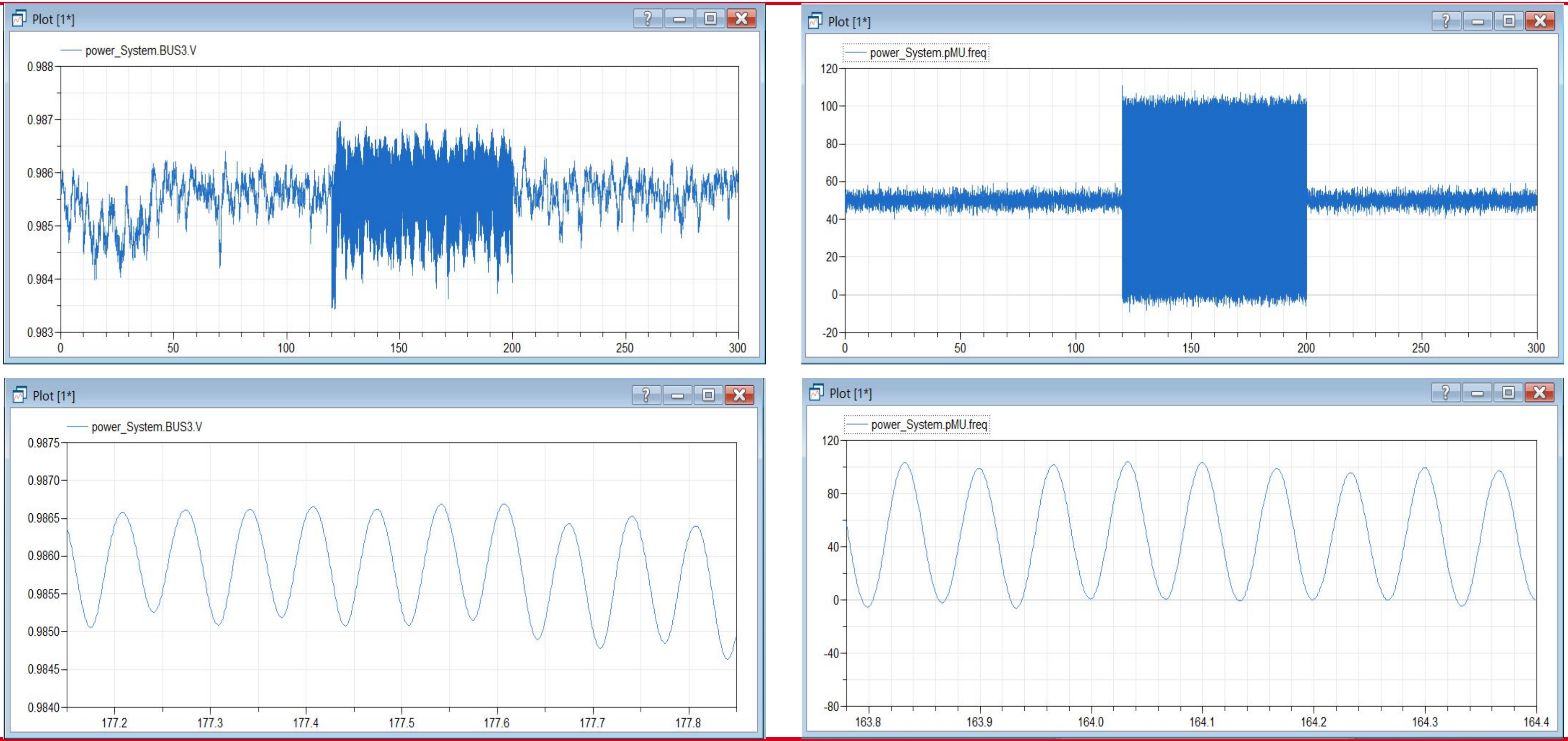
Construct the Replaceable Model



Propagate Parameters



Simulate the Model in Dymola



Data Generation and Model Training in Python

Rensselaer

Simulate the Model Using Dymola Python Interfaces

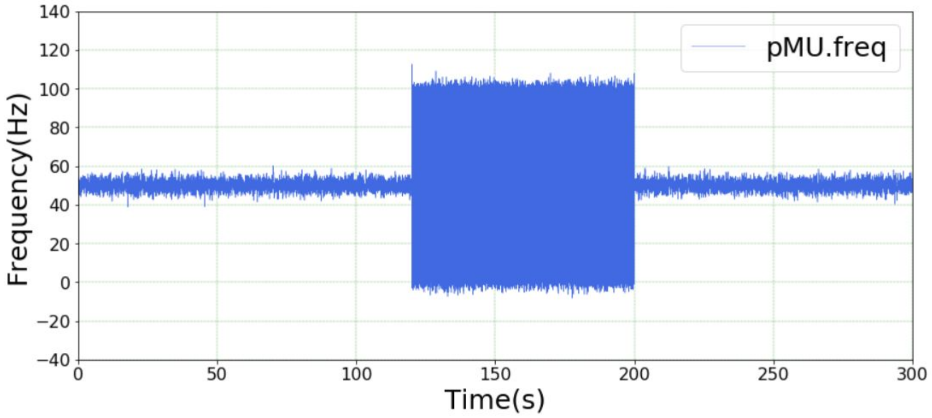
```
import os
import sys
sys.path.insert(0, os.path.join('E:\\', 'Spring2021', 'MnS4CPS', 'Dymola', 'Modelica',
                                'Library', 'python_interface', 'dymola.egg'))
```

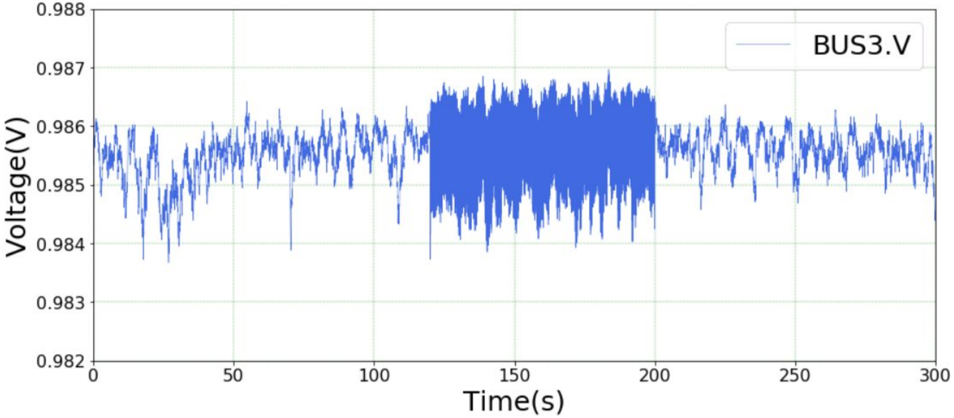
- Insert the path of the "dymola.egg"

```
dymola = DymolaInterface() # Instantiate the Dymola interface and start Dymola
dymola.openModel("E:/Spring2021/MnS4CPS/Final Project/2018_AmericanModelicaConf_PowerGrid_plus_PowerSystems/Modelica_Models/Open
dymola.cd("E:/Spring2021/MnS4CPS/Final Project/") # change the working directory
dymola.simulateExtendedModel("OpenCPS_D53B.MLTraining.Configurations.SMIB_normal_noise_signalB_test",
                    startTime = 0,
                    stopTime = 300,
                    outputInterval = 0.001,
                    method = "Radau",
                    tolerance = 0.0001,
                    resultFile = 'MLtraining',
                    initialNames = ["signal.sine.amplitude", "signal.greaterThreshold.threshold", "signal.greaterThreshold1.thres
                    initialValues = [osci_amp, osci_start, osci_end],
                    finalNames = ["signal.sine.amplitude", "signal.greaterThreshold.threshold", "signal.greaterThreshold1.thresh
num_of_rows = dymola.readTrajectorySize("MLtraining.mat")
data = dymola.readTrajectory("MLtraining.mat", ["Time", "power_System.pMU.freq", "power_System.BUS3.V"], num_of_rows)
time = data[0]
PMU_Freq = data[1]
BUS3_V = data[2]
```

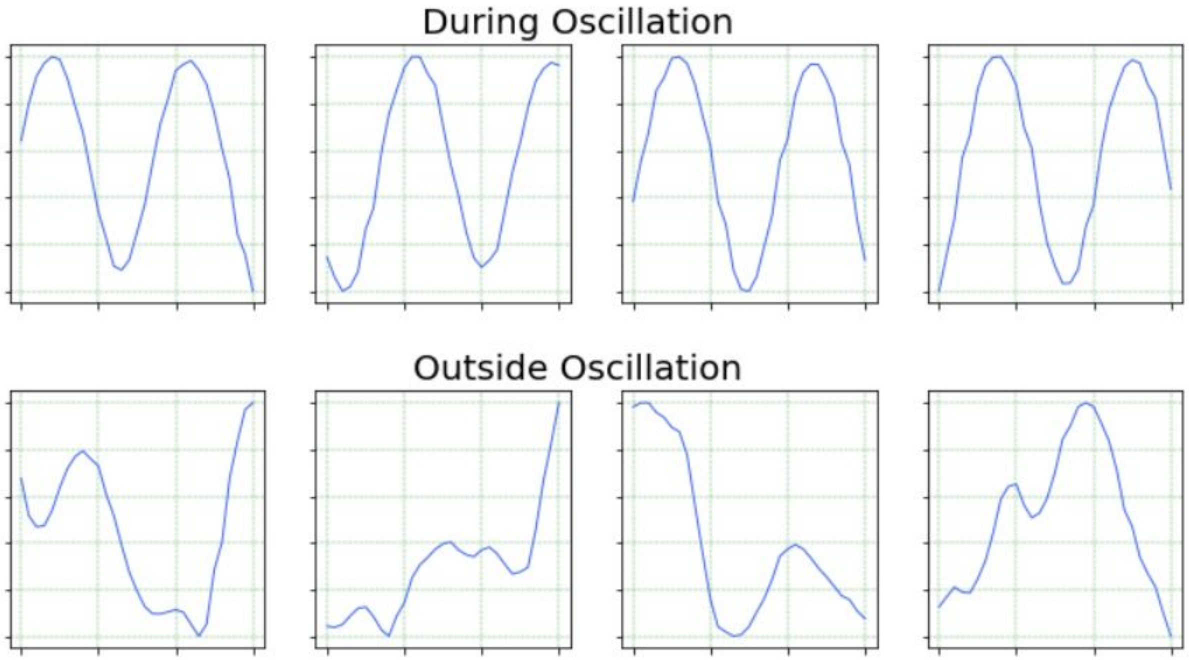
- Instantiating the Dymola interface
- simulateExtendedMode function
- readTrajectory function

Simulate the Model Using Dymola Python Interfaces



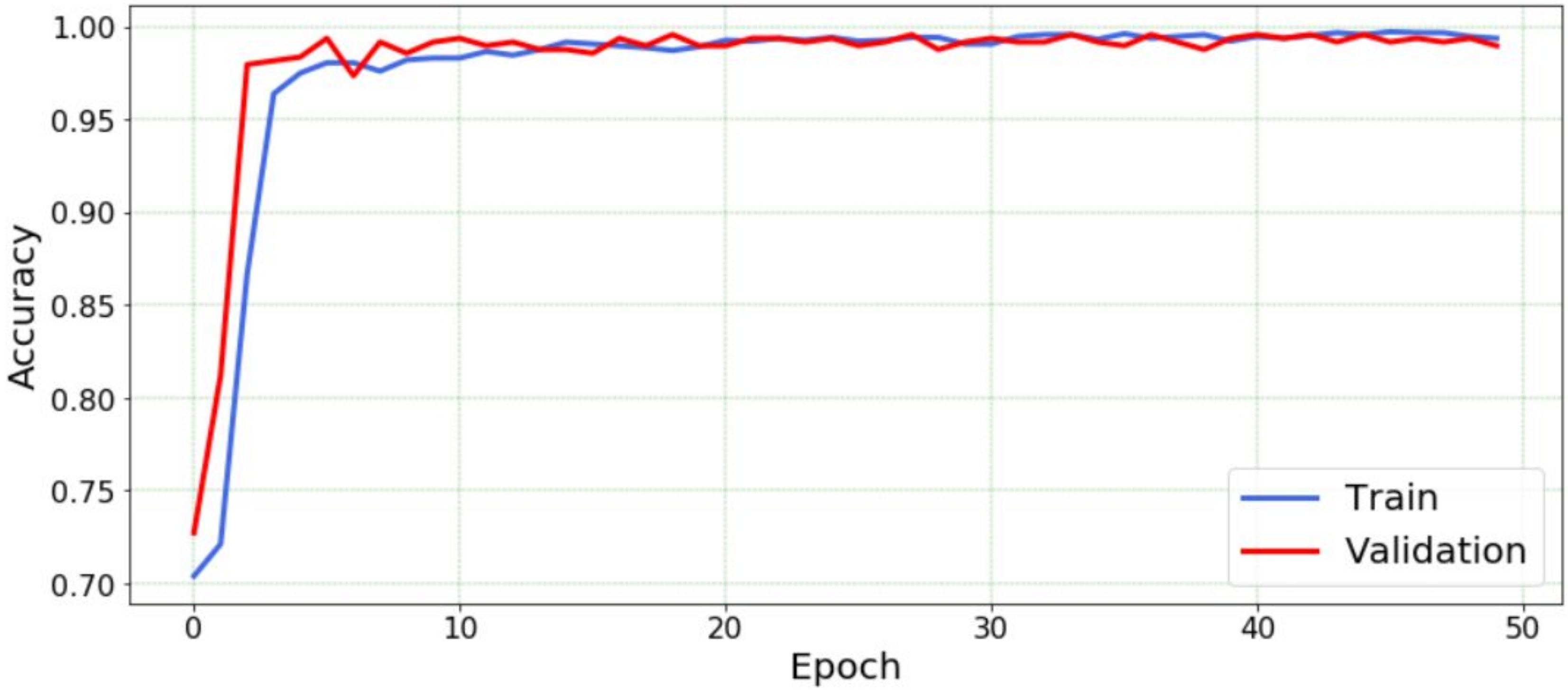


Data Generation Using Dymola Python Interfaces

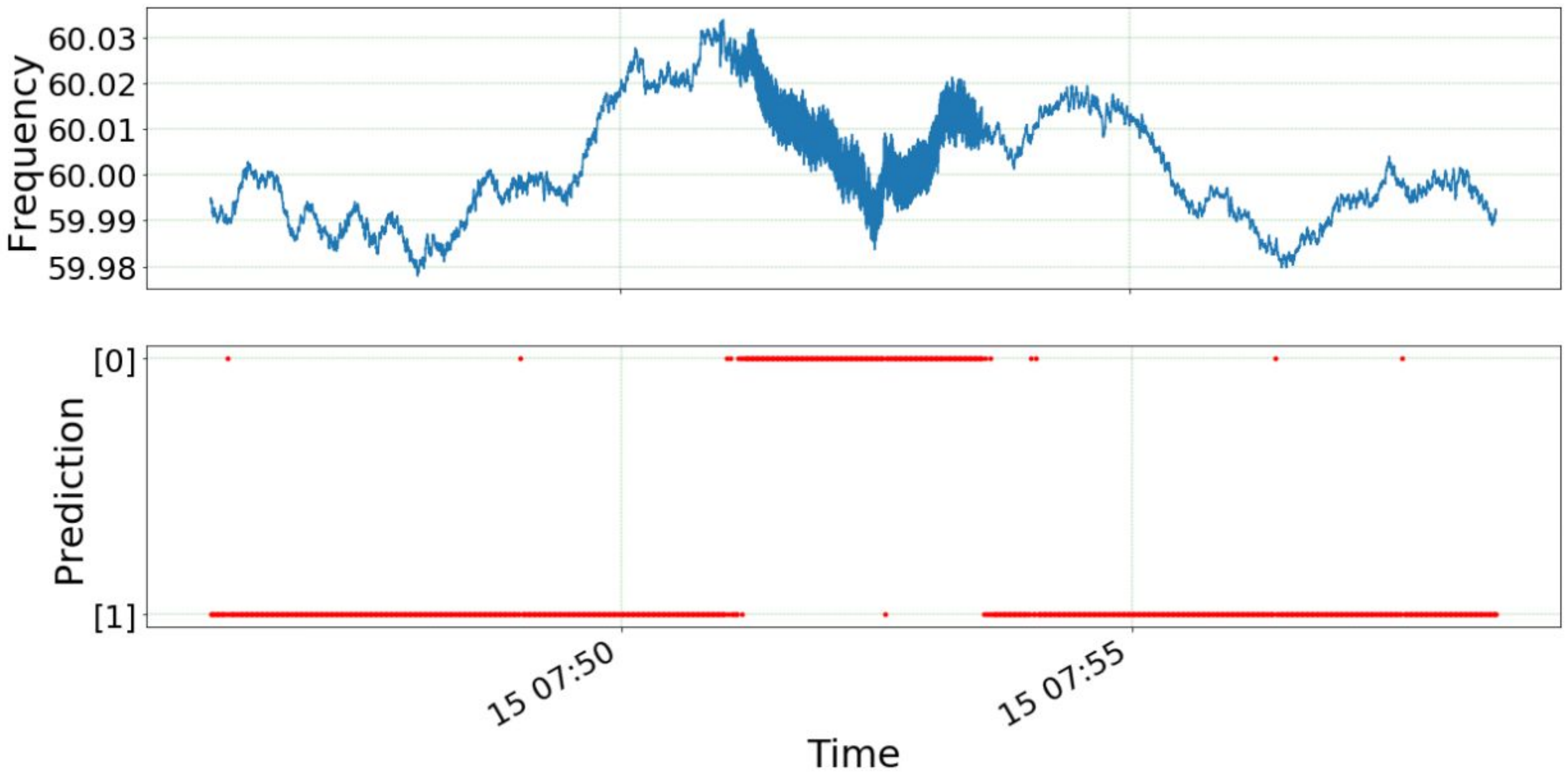


- Reshape the generated dataset into small groups for ML training
- Label each group with 1 and 0

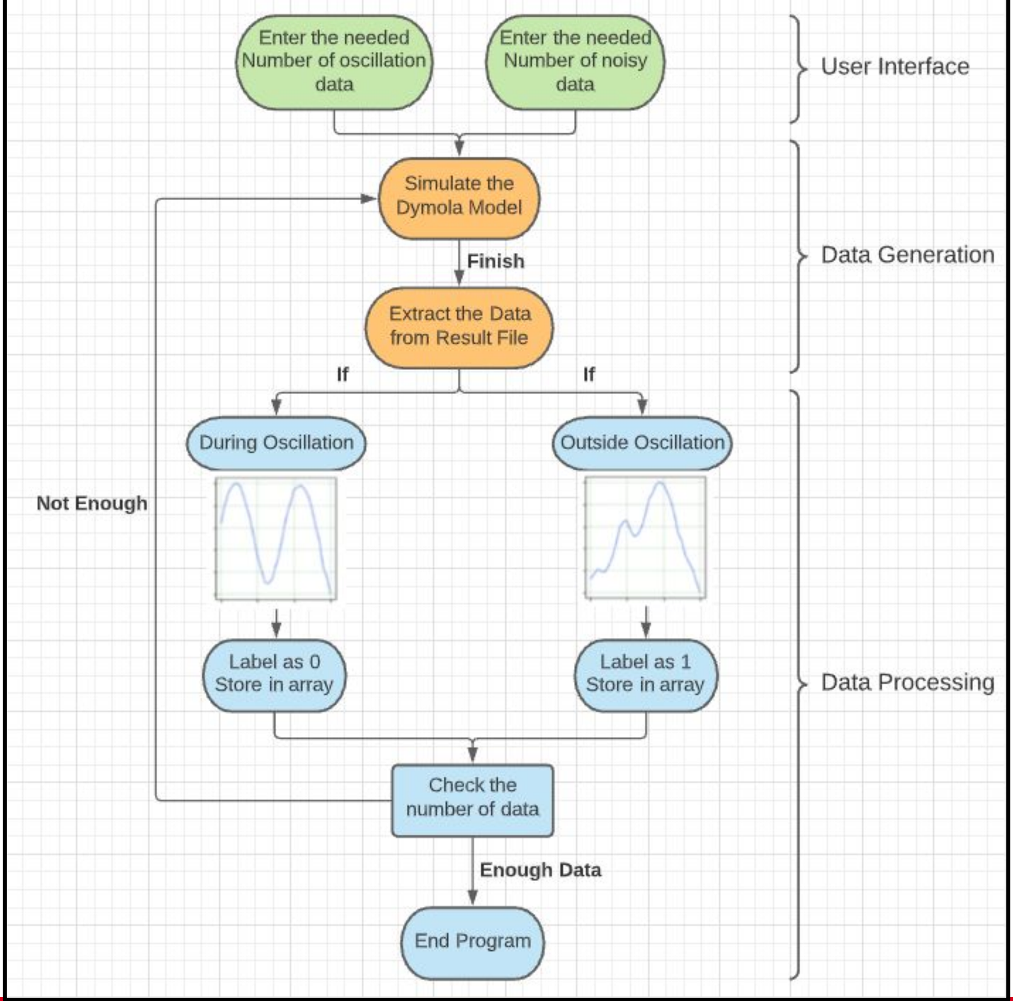
Training of the ML Model



Prediction Result of the Model Trained by Dymola Data



Automatic Data Generation



Workflow of the Automatic Data Generation Program

- User Interface
  - Enter the number
- Data Generation
  - Simulation
  - Data Extraction
- Data Processing
  - Reshape
  - Labeling
  - Check the number

Automatic Data Generation

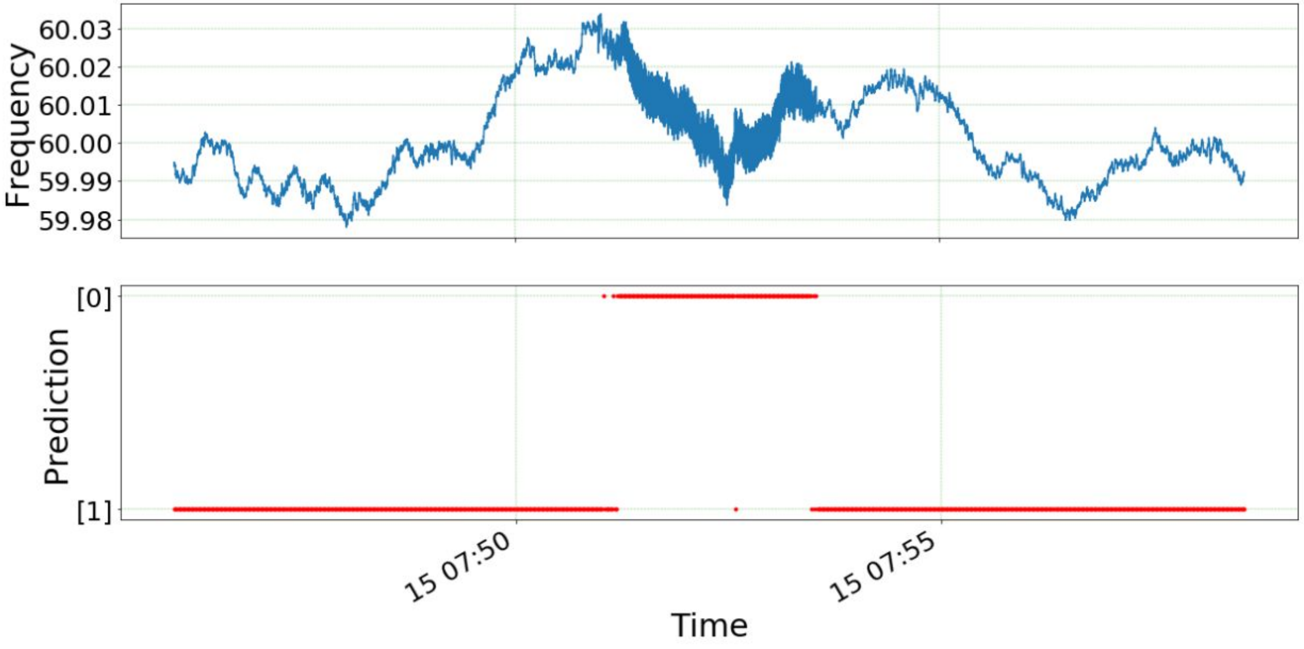
```
# Set the Parameters of the Dymola model
osci_amp = 4              # Oscillation Amplitude
osci_start = 120          # Oscillation Start Time
osci_end = 200            # Oscillation End Time
simulation_end = 300      # Simulation End Time

# Set the amount of data that need to be generated
osci_data_num = 1000                             that needs to be generated
nois_data_num = 1000      # Number of noisy data that needs to be generated
```

Enter the number of noisy data that needs to be generated, e.g. 1000.

Enter the number of oscillation data that needs to be generated, e.g. 1000.

Prediction Result of the Model Trained by Dymola Data



Num of Oscillation Data = 4000

Num of Noisy Data = 4000

Total Inferences = 756

Number of Error = 2

Accuracy = 0.99735

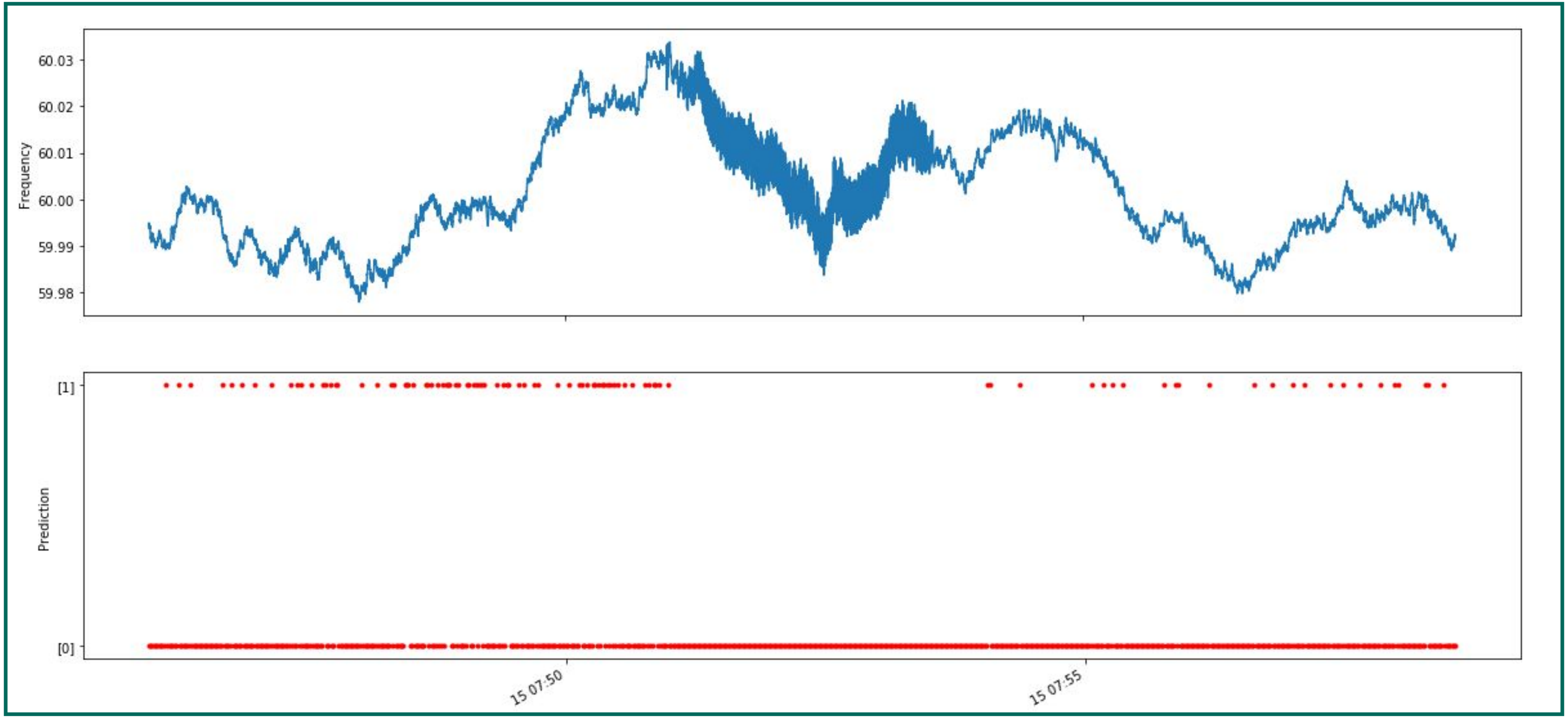
Experiments by Varying the Number of Data

ALSET lab

Num of Oscillation Data	Num of Noisy Data	Num of Error	Accuracy
1000	1000	16	0.97884
2000	2000	14	0.98148
3000	3000	7	0.99074
4000	4000	2	0.99735

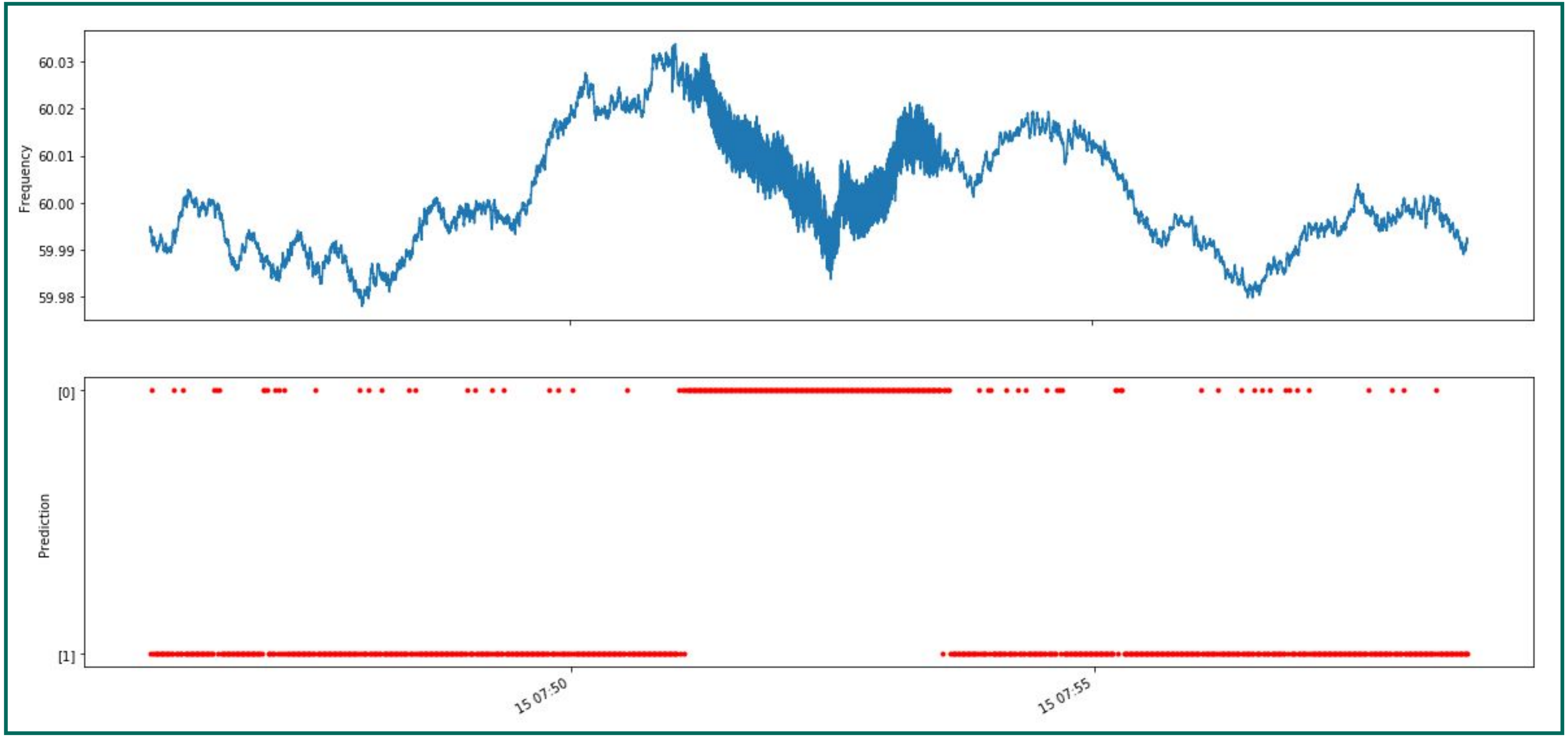
Compare with Previous Models

Prediction Result of the Model Trained by Python Data



Prediction Result of the Model Trained by Analog Discovery Board Data

ALSET lab



Conclusion

- The model is a very good approximation of the power system
- The model is constructed with replaceable structure
- Successful implementation of Dymola Python API to generate ML training data from the simulation
- Data Generation Process is automatic
- Data Generated by Dymola model is close-to-reality
  - Better than Python Data and Analog Discovery Data
- Accuracy of the ML model is increased by generating more data