EE 523 Power System Stability and Control

Final Report

Small Signal Analysis and Transient Stability Analysis of Type 2 and Type 3 System

May 2024

Submitted by

Athul Jose P
11867566

School of Electrical Engineering and Computer Science
Washington State University

Contents

List of Figures

List of Tables

1 Introduction

This report investigates the small signal and transient stability characteristics of a power system using the Kundur two-area model. Power system stability is crucial for maintaining reliable and secure operation in electrical grids. Small signal stability analysis focuses on the response of the system to small disturbances, while transient stability analysis assesses its ability to withstand large disturbances and maintain synchronism.

The Kundur two-area system provides a simplified representation of a power system with two interconnected areas, synchronous generators, and loads. This model facilitates the study of dynamic interactions between different components and the effectiveness of control measures in maintaining stability. This report provides an overview of the Kundur two-area model and the mathematical formulation of system dynamics. It discusses methodologies for small signal and transient stability analysis. Simulations of both Type 2 and Type 3 models are conducted to compare stability characteristics. Furthermore, Power System Stabilizers (PSS) are implemented in the Type 2 model to assess their impact on system stability.

Results from simulations are analyzed to understand system responses under various operating conditions and disturbance scenarios. By studying Kundur two-area system stability with PSS implementation, insights into dynamic behavior and control strategies are gained, contributing to the development of robust power system designs.

2 System Model

2.1 Kundur 2 Area System

The Kundur two-area system [1] is a classic example often used in power system stability and control studies. It consists of two interconnected synchronous generators, each representing an area with its own load. This simplified model serves as a test bed for analyzing various aspects of power system dynamics, including transient stability, small-signal stability, and control strategies. Studying the Kundur two-area system helps engineers and researchers understand the behavior of power systems under different operating conditions and disturbances, facilitating the development of effective control and protection schemes for real-world power grids.

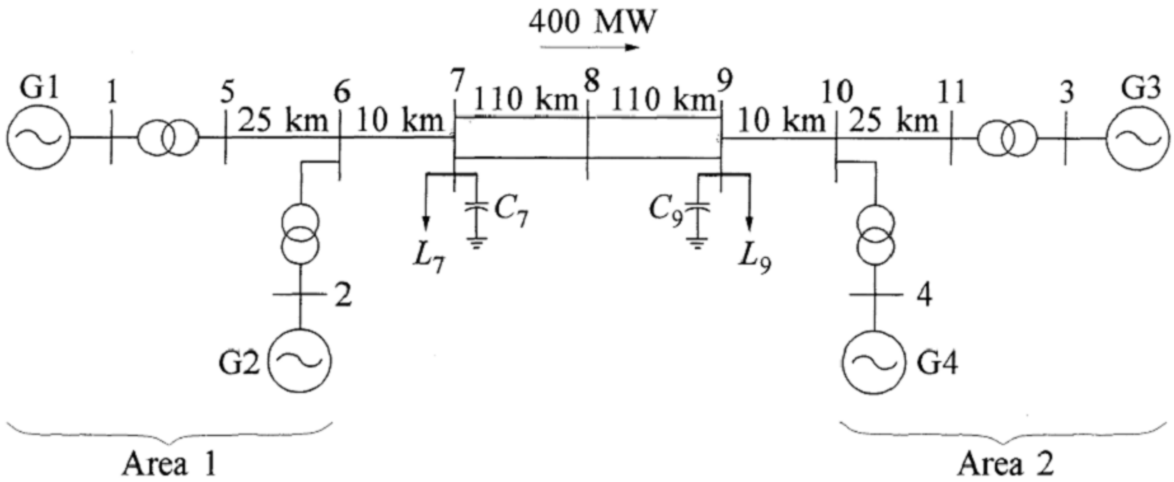

Figure 1: Kundur 2 area model

2.2 Type 2 Modeling

Type 2 modeling is an advancement over Type 1 system models, offering a balance between accuracy and computational efficiency. While retaining essential details of machine dynamics, Type 2 models streamline simulation time by employing network reduction techniques. This reduction involves certain assumptions:

1. **Machine Saliency Ignored**
   Synchronous machine saliency effects are disregarded, focusing on simplifying the network without compromising overall accuracy.

2. **Constant Impedance Loads**
   Loads in the system are assumed to be of constant impedance type, facilitating easier network reduction and analysis.

The system model resulting from Type 2 modeling incorporates these simplifications while preserving essential machine and network dynamics. The model serves as a foundation for analyzing transient stability, dynamic response, and control strategies in power systems.

Swing equations

$$\dot{\theta}_i = (\omega_i - 1) \cdot \omega_s \tag{1}$$

$$\dot{\omega}_i = \frac{1}{2H_i} \left(P_{m_i} - P_{e_i} - K_{D_i}(\omega_i - 1)\right) \tag{2}$$

with

$$P_{e_i} = \sum_{j=1}^{N_{G+1}} Y_{G_{ij}} E_i' E_j' \cos(\gamma_i - \gamma_j - \theta_{G_{ij}})$$

Electromagnetic equations

$$\dot{E}_{q_i} = \frac{1}{T_{do_i}} \left(-E_{q_i}' - (X_{d_i} - X_{d_i}')I_{d_i} + E_{fd_i}\right) \tag{3}$$

$$\dot{E}_{d_i} = \frac{1}{T_{qo_i}} \left(-E_{d_i}' + (X_{q_i} - X_{q_i}')I_{q_i}\right) \tag{4}$$

2.3 Excitation System

The excitation system in a power system plays a pivotal role in regulating the voltage output of synchronous generators, ensuring stability and reliability in electrical grids. It functions by adjusting the field current supplied to the generator's rotor windings, thereby controlling the magnitude and phase angle of the terminal voltage. This regulation is critical for maintaining system stability, especially during transient conditions and load fluctuations. Modern excitation systems are equipped with advanced control algorithms and feedback mechanisms to swiftly respond to changes in operating conditions and grid disturbances. By maintaining voltage levels within predefined limits, excitation systems contribute significantly to grid stability, power quality, and overall system performance.

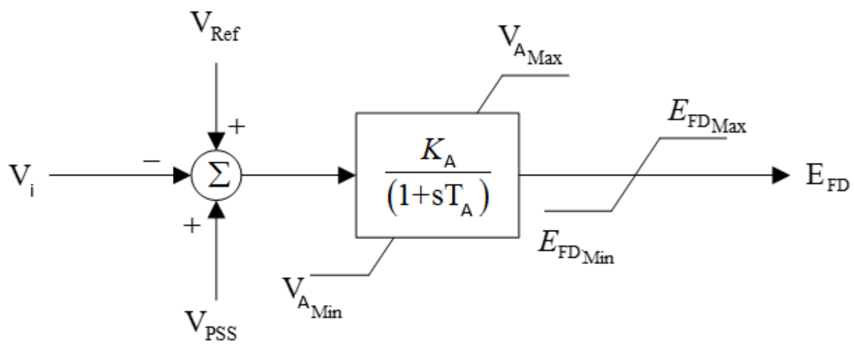

Figure 2: Block diagram of excitation system

$$\dot{E_{fdi}} = \frac{1}{T_{Ai}}\left(-E_{fdi} + K_{Ai}(V_{\text{ref}_i} - V_i)\right) \tag{5}$$

2.4 Speed Governor

The speed governor of a synchronous generator is an essential component that regulates the rotational speed of the generator's prime mover, typically a steam turbine or a hydro turbine. Its primary function is to maintain the generator's output frequency at the desired value, ensuring synchronization with the grid and stable operation. The governor achieves this by controlling the flow of fuel or water to the prime mover in response to changes in electrical load or grid conditions. By adjusting the turbine's output power, the governor effectively controls the rotational speed of the generator, helping to maintain grid stability and frequency within acceptable limits. Modern speed governors are equipped with sophisticated control algorithms and feedback mechanisms to provide fast and accurate response to changes in load and system conditions, thereby ensuring reliable and efficient operation of the power system.

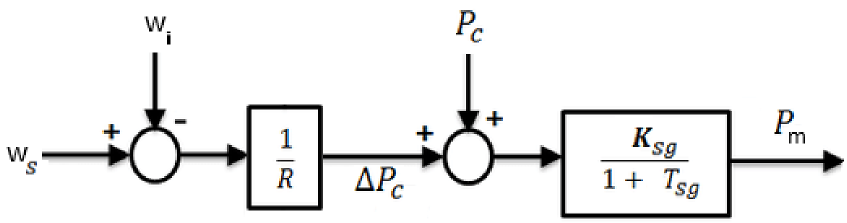

Figure 3: Block diagram of speed governor

$$\dot{P}_{m_i} = \frac{1}{T_{sg_i}}\left(-P_{m_i} + P_{c_i} + \frac{1}{R_i}(1 - w_i)\right) \tag{6}$$

Additional equations required to setup the system is given by Coupling equations

$$E'_i = \sqrt{E_{d_i}^2 + E_{q_i}^2}$$
$$E'_{d_i} = V_{d_i} + R_{s_i} I_{d_i} - X'_i I_{q_i}$$
$$E'_{q_i} = V_{q_i} + R_{s_i} I_{q_i} + X'_i I_{d_i}$$

with

$$I_{d_i} = \sum_{j=1}^{N_{G+1}} Y_{G_{ij}} E'_j \sin(\gamma_i - \gamma_j - \theta_{G_{ij}})$$
$$I_{q_i} = \sum_{j=1}^{N_{G+1}} Y_{G_{ij}} E'_j \cos(\gamma_i - \gamma_j - \theta_{G_{ij}})$$

2.5 Power System Stabilizer

Power System Stabilizers (PSS) are control devices used in power systems to improve stability by providing supplementary control signals to synchronous generators. These devices help mitigate low-frequency oscillations and enhance the damping of system modes, thereby ensuring the stability of the power grid. PSS operates by measuring the speed deviation of the generator rotor from its nominal speed and generating a supplementary control signal to adjust the generator excitation. By modulating the excitation system, PSS can effectively dampen out oscillations caused by disturbances such as sudden load changes or faults. The dynamic equations of PSS is given by

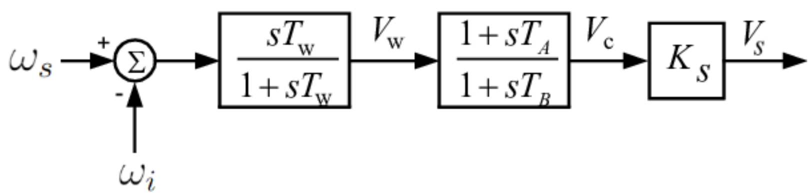

Figure 4: Block diagram of power system stabilizer

$$\dot{V}_{w_i} = \frac{1}{T_{w_i}} \left(-V_{w_i} - T_{w_i}\dot{\omega}_i\right) \tag{7}$$
$$\dot{V}_{s_i} = \frac{1}{K_{s_i} T_{b_i}} \left(-K_{s_i} V_{w_i} + V_{w_i} + T_{a_i}\dot{V}_{w_i}\right) \tag{8}$$

with

$$\dot{E_{fdi}} = \frac{1}{T_{Ai}} \left(-E_{fdi} + K_{Ai}(V_{\text{ref}_i} + V_{s_i} - V_i) \right)$$

2.6 Type 3 Modeling

Type 3 modeling builds upon the simplifications of Type 2, introducing additional assumptions to further streamline the representation of synchronous generators. In this model, it is assumed that the generator's internal voltage remains constant, effectively disregarding the dynamics associated with changes in this parameter. Consequently, the number of state equations is reduced to just 2, significantly enhancing simulation efficiency, particularly for transient analyses. By sacrificing some level of detail regarding internal voltage dynamics, Type 3 models strike a balance between computational complexity and simulation accuracy, making them suitable for various power system stability studies. The equations involved are:

Swing equations

$$\dot{\theta}_i = (\omega_i - 1) \cdot \omega_s \tag{9}$$

$$\dot{\omega}_i = \frac{1}{2H_i} \left(P_{m_i} - P_{e_i} - K_{D_i}(\omega_i - 1) \right) \tag{10}$$

with

$$P_{e_i} = \sum_{j=1}^{N_{G+1}} Y_{G_{ij}} E'_i E'_j \cos(\theta_i - \theta_j - \theta_{G_{ij}})$$

3 Simulation Setup

Kundur system used for simulation has been modified from original test system. The lines between 6-7, 7-8, 8-9 and 9-10 are doubled. The system of equations are modelled in MATLAB environment [2]. The system data is entered through a cdf file. The information of the cdf file is given below: The analysis consists of three parts which are given below:-

1	Bus 1	HV	1	1	3	1.030	20.20	0.0	0.0	700.0	185.0	0.0	1.030	0.0	0.0	0.0	0.0	0
2	Bus 2	HV	1	1	0	1.010	10.50	0.0	0.0	700.0	235.0	0.0	1.010	0.0	0.0	0.0	0.0	0
3	Bus 3	HV	2	1	0	1.030	-6.80	0.0	0.0	719.0	176.0	0.0	1.030	0.0	0.0	0.0	0.0	0
4	Bus 4	HV	2	1	0	1.010	-17.00	0.0	0.0	700.0	202.0	0.0	1.010	0.0	0.0	0.0	0.0	0
5	Bus 5	HV	1	1	0	1.006	13.74	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
6	Bus 6	LV	1	1	0	0.978	3.65	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
7	Bus 7	ZV	1	1	0	0.961	-4.76	967.0	100.0	0.0	200.0	0.0	1.000	0.0	0.0	9.5039	0.9828	0
8	Bus 8	TV	3	1	0	0.949	-18.64	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
9	Bus 9	LV	2	1	0	0.971	-32.24	1767.0	250.0	0.0	350.0	0.0	1.000	0.0	0.0	17.1651	2.4286	0
10	Bus 10	LV	2	1	0	0.984	-23.82	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
11	Bus 11	LV	2	1	0	1.008	-13.51	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
12	Bus 12	LV	1	1	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
13	Bus 13	LV	1	1	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
14	Bus 14	LV	1	1	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0

Figure 5: Kundur 2 area system bus data

BRANCH DATA FOLLOWS

1	5	1	1	1	0	0.00000	0.01667	0.0	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	6	1	1	1	0	0.0	0.01667	0.0	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	12	1	1	1	0	0.00000	0.04722	0.0	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	11	2	1	1	0	0.0	0.01667	0.0	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	13	2	1	1	0	0.00000	0.04722	0.0	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	10	2	1	1	0	0.00000	0.01667	0.0	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	14	2	1	1	0	0.0	0.04722	0.0	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	6	1	1	1	0	0.00250	0.02500	0.04375	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	7	1	1	1	0	0.00050	0.00500	0.03500	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	8	1	1	1	0	0.00275	0.02750	0.77000	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	9	2	1	1	0	0.00275	0.02750	0.77000	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	10	2	1	1	0	0.00050	0.00500	0.03500	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	11	2	1	1	0	0.00250	0.02500	0.04375	0	0	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 6: Kundur 2 area system branch data

3.1 Dynamic Initialization

To commence simulations accurately, the system undergoes initialization with standard parameter values. Equations are then solved iteratively until reaching a state where all equations equate to zero. This process yields the initial conditions necessary for a stable system state. Through this iterative solving approach, the system attains equilibrium, ensuring precise simulation outcomes.

3.2 Small Signal Stability

This segment delves into scrutinizing the system's small signal stability. Initially, the Jacobian matrix of the system is computed to examine its linearized behavior around an equilibrium point. Following this, the dynamic initialization outcomes are integrated into the Jacobian to derive the system's eigenvalues. It is imperative for small signal stability that all eigenvalues exhibit negative real values, indicating system stability under small disturbances.

3.3 Transient Stability

This section involves conducting transient simulations to assess the system's behavior under various conditions. Distinct sets of equations are formulated for pre-fault, fault-on, and post-fault scenarios, each reflecting the system's dynamics at different stages. The simulations commence from the dynamic initialization results, utilizing Euler integration to calculate the subsequent step values. Subsequently, the generator angles (thetas) and speeds (omegas) are graphed to evaluate the system's stability. For this transient stability analysis, a fault is introduced midway between buses 6 and 7. Multiple simulations are conducted by adjusting the clearing time to identify the critical clearing time, which indicates the minimum time required to clear the fault while maintaining stability. The data for the fault and post fault scenario is give through the .cdf files. They are given as follows:-

1	Bus 1	HV	1	1	3	1.030	20.20	0.0	0.0	700.0	185.0	0.0	1.030	0.0	0.0	0.0	0.0	0
2	Bus 2	HV	1	1	0	1.010	10.50	0.0	0.0	700.0	235.0	0.0	1.010	0.0	0.0	0.0	0.0	0
3	Bus 3	HV	2	1	0	1.030	-6.80	0.0	0.0	719.0	176.0	0.0	1.030	0.0	0.0	0.0	0.0	0
4	Bus 4	HV	2	1	0	1.010	-17.00	0.0	0.0	700.0	202.0	0.0	1.010	0.0	0.0	0.0	0.0	0
5	Bus 5	HV	1	1	0	1.006	13.74	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
6	Bus 6	LV	1	1	0	0.978	3.65	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	19.8019	-198.0198	0
7	Bus 7	ZV	1	1	0	0.961	-4.76	967.0	100.0	0.0	200.0	0.0	1.000	0.0	0.0	29.3058	-197.0282	0
8	Bus 8	TV	3	1	0	0.949	-18.64	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
9	Bus 9	LV	2	1	0	0.971	-32.24	1767.0	250.0	0.0	350.0	0.0	1.000	0.0	0.0	17.1651	2.4286	0
10	Bus 10	LV	2	1	0	0.984	-23.82	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
11	Bus 11	LV	2	1	0	1.008	-13.51	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
12	Bus 12	LV	1	1	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
13	Bus 13	LV	1	1	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0
14	Bus 14	LV	1	1	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.0	0.0	0.0	0

Figure 7: Kundur 2 area system bus data with fault

```
BRANCH DATA FOLLOWS
   1    5  1  1 1 0  0.00000    0.01667    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
  12    6  1  1 1 0  0.0        0.01667    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   2   12  1  1 1 0  0.00000    0.04722    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
  13   11  2  1 1 0  0.0        0.01667    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   3   13  2  1 1 0  0.00000    0.04722    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
  14   10  2  1 1 0  0.00000    0.01667    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   4   14  2  1 1 0  0.0        0.04722    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   5    6  1  1 1 0  0.00250    0.02500    0.04375    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   6    7  1  1 1 0  0.00100    0.01000    0.01750    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   7    8  1  1 1 0  0.00275    0.02750    0.77000    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   8    9  2  1 1 0  0.00275    0.02750    0.77000    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   9   10  2  1 1 0  0.00050    0.00500    0.03500    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
  10   11  2  1 1 0  0.00250    0.02500    0.04375    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
```

Figure 8: Kundur 2 area system branch data with fault

```
  1 Bus 1     HV  1  1  3 1.030  20.20     0.0      0.0    700.0   185.0     0.0  1.030    0.0    0.0   0.0      0.0       0
  2 Bus 2     HV  1  1  0 1.010  10.50     0.0      0.0    700.0   235.0     0.0  1.010    0.0    0.0   0.0      0.0       0
  3 Bus 3     HV  2  1  0 1.030  -6.80     0.0      0.0    719.0   176.0     0.0  1.030    0.0    0.0   0.0      0.0       0
  4 Bus 4     HV  2  1  0 1.010 -17.00     0.0      0.0    700.0   202.0     0.0  1.010    0.0    0.0   0.0      0.0       0
  5 Bus 5     HV  1  1  0 1.006  13.74     0.0      0.0      0.0     0.0     0.0  1.000    0.0    0.0   0.0      0.0       0
  6 Bus 6     LV  1  1  0 0.978   3.65     0.0      0.0      0.0     0.0     0.0  1.000    0.0    0.0   0.0      0.0       0
  7 Bus 7     ZV  1  1  0 0.961  -4.76   967.0    100.0      0.0   200.0     0.0  1.000    0.0    0.0   9.5039   0.9828    0
  8 Bus 8     TV  3  1  0 0.949 -18.64     0.0      0.0      0.0     0.0     0.0  1.000    0.0    0.0   0.0      0.0       0
  9 Bus 9     LV  2  1  0 0.971 -32.24  1767.0    250.0      0.0   350.0     0.0  1.000    0.0    0.0  17.1651   2.4286    0
 10 Bus 10    LV  2  1  0 0.984 -23.82     0.0      0.0      0.0     0.0     0.0  1.000    0.0    0.0   0.0      0.0       0
 11 Bus 11    LV  2  1  0 1.008 -13.51     0.0      0.0      0.0     0.0     0.0  1.000    0.0    0.0   0.0      0.0       0
 12 Bus 12    LV  1  1  0 0.0    0.0       0.0      0.0      0.0     0.0     0.0  1.000    0.0    0.0   0.0      0.0       0
 13 Bus 13    LV  1  1  0 0.0    0.0       0.0      0.0      0.0     0.0     0.0  1.000    0.0    0.0   0.0      0.0       0
 14 Bus 14    LV  1  1  0 0.0    0.0       0.0      0.0      0.0     0.0     0.0  1.000    0.0    0.0   0.0      0.0       0
```

Figure 9: Kundur 2 area system bus data data for post fault

```
BRANCH DATA FOLLOWS
   1    5  1  1 1 0  0.00000    0.01667    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
  12    6  1  1 1 0  0.0        0.01667    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   2   12  1  1 1 0  0.00000    0.04722    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
  13   11  2  1 1 0  0.0        0.01667    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   3   13  2  1 1 0  0.00000    0.04722    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
  14   10  2  1 1 0  0.00000    0.01667    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   4   14  2  1 1 0  0.0        0.04722    0.0        0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   5    6  1  1 1 0  0.00250    0.02500    0.04375    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   6    7  1  1 1 0  0.00100    0.01000    0.01750    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   7    8  1  1 1 0  0.00275    0.02750    0.77000    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   8    9  2  1 1 0  0.00275    0.02750    0.77000    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
   9   10  2  1 1 0  0.00050    0.00500    0.03500    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
  10   11  2  1 1 0  0.00250    0.02500    0.04375    0     0     0    0 0  0.0        0.0 0.0   0.0     0.0    0.0   0.0
```

Figure 10: Kundur 2 area system branch data for post fault

4 Simulation Results

4.1 Type 3 Model

The Type 3 model entails two state equations per generator, resulting in a total of six state equations for three generators. The system of equations is solved using MATLAB's "vpasolve" function, with inputs directly extracted from the .cdf files. The simulation process comprises dynamic initialization, followed by small signal stability and transient stability analyses.

4.1.1 Dynamic Initialization

The system equations are initialized with typical values, and then solved to obtain the dynamic initialization state. The resulting initialized condition is depicted below.

Steady state value	State
0.154137806	θ_2
0.117836646	θ_3
-0.043987884	θ_4
1.0	ω_2
1.0	ω_3
1.0	ω_4

Table 1: Dynamic Initialization of Type 3 model

4.1.2 Small Signal Stability

The dynamic initialization is applied to the Jacobian matrix to calculate the eigenvalues. Analysis of the eigenvalues confirms that all eigenvalues possess a negative real part, indicating system stability under small signal conditions. However, the observed damping in the system is minimal, suggesting the need for Power System Stabilizers (PSS). Nevertheless, it's worth noting that Type 3 modeling, which assumes constant generator internal voltages, does not support the inclusion of PSS.

Table 3 showcases the participation matrix, delineating the impact of each state on various modes. A participation factor of 1 designates the predominant state for the

Mode	Eigen Values	Frequency	Damping	Mode Type
1	-0.008966 + 3.1548i	0.5021	0.2842	Intra
2	-0.008966 - 3.1548i	0.5021	0.2842	Intra
3	-0.0089898 + 6.6678i	1.0612	0.1348	Local
4	-0.0089898 - 6.6678i	1.0612	0.1348	Local
5	-0.0085849 + 5.844i	0.9301	0.1469	Intra
6	-0.0085849 - 5.844i	0.9301	0.1469	Intra

Table 2: Eigen values of Type 3 model

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
θ_2	0.1237	0.1237	0.0255	0.0255	1	1
θ_3	1	1	0.5927	0.5927	0.0877	0.0877
θ_4	0.6833	0.6833	1	1	0.0042	0.0042
ω_2	0.1237	0.1237	0.0255	0.0255	1	1
ω_3	1	1	0.5927	0.5927	0.0877	0.0877
ω_4	0.6833	0.6833	1	1	0.0042	0.0042

Table 3: Participation matrix for Type 3 model

respective mode. For instance, in mode 1, θ_2 and ω_2 emerge as the dominant states.

4.1.3 Transient Stability

Transient analysis involves simulating a fault on the transmission line connecting buses 6 and 7, with varying fault clearing durations. These simulations are repeated iteratively until the system reaches instability, allowing for the observation of system behavior under different fault-clearing scenarios.

The simulation commenced with a clearing time set at 3 cycles, with subsequent simulations conducted by incrementing the clearing time by one cycle. Analysis of the results revealed that the system failed to recover from the disturbance at the 7-cycle clearing time mark. Therefore, the critical clearing time was determined to be 6 cycles.

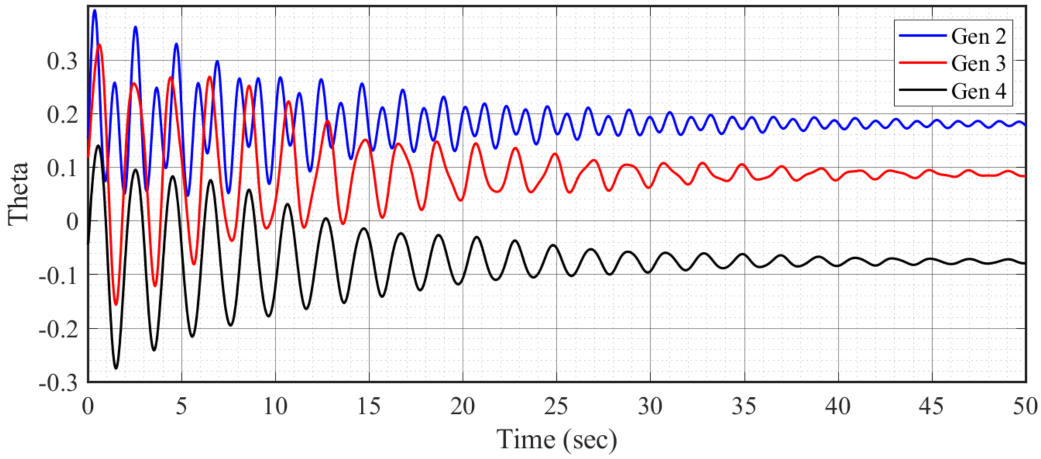

Figure 11: Generator angle vs time when Tc = 3 cycles

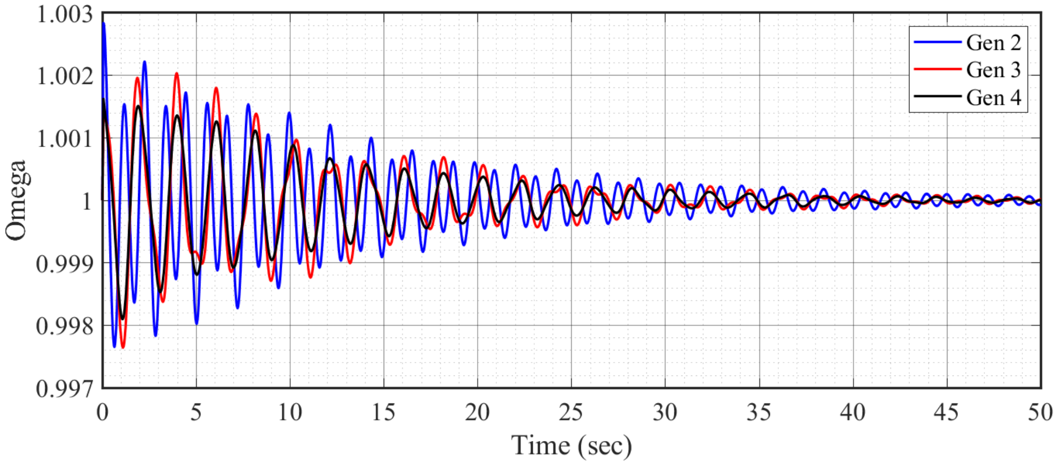

Figure 12: Generator speed vs time when Tc = 3 cycles

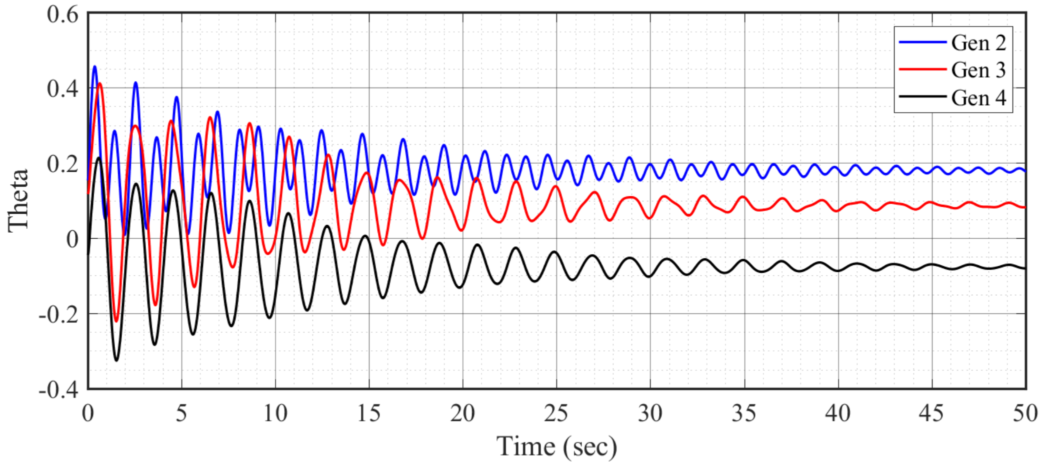

Figure 13: Generator angle vs time when Tc = 4 cycles

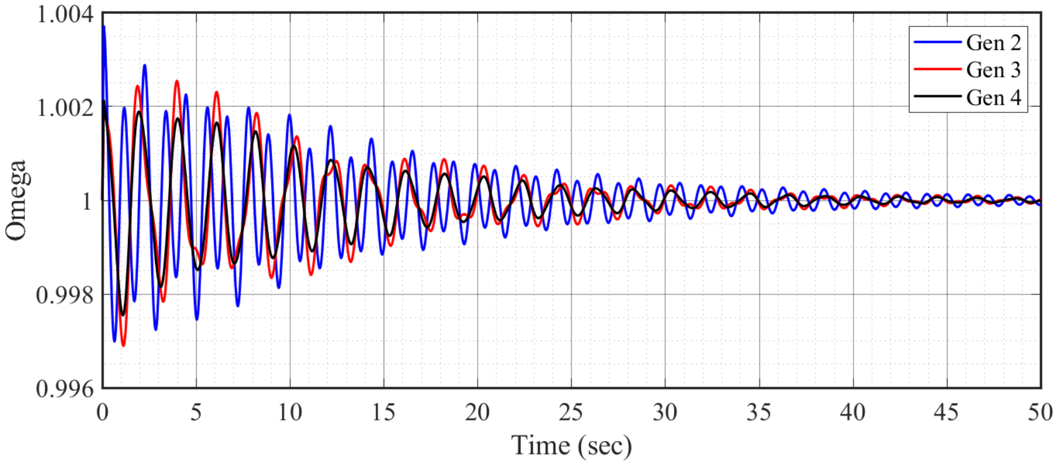

Figure 14: Generator speed vs time when Tc = 4 cycles

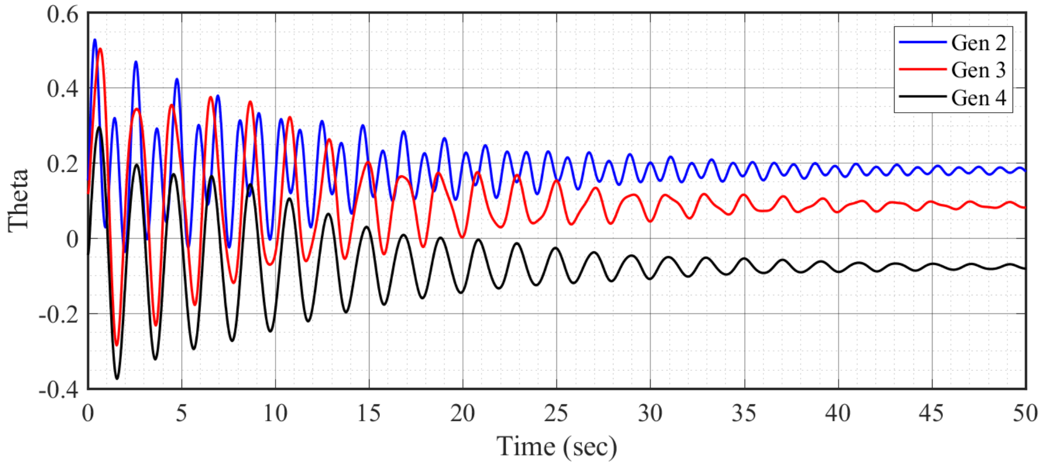

Figure 15: Generator angle vs time when Tc = 5 cycles

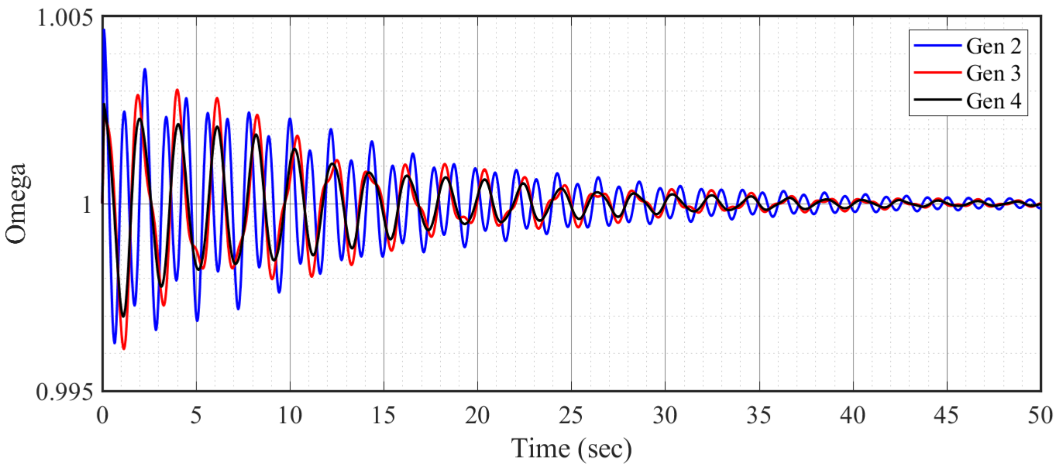

Figure 16: Generator speed vs time when Tc = 5 cycles

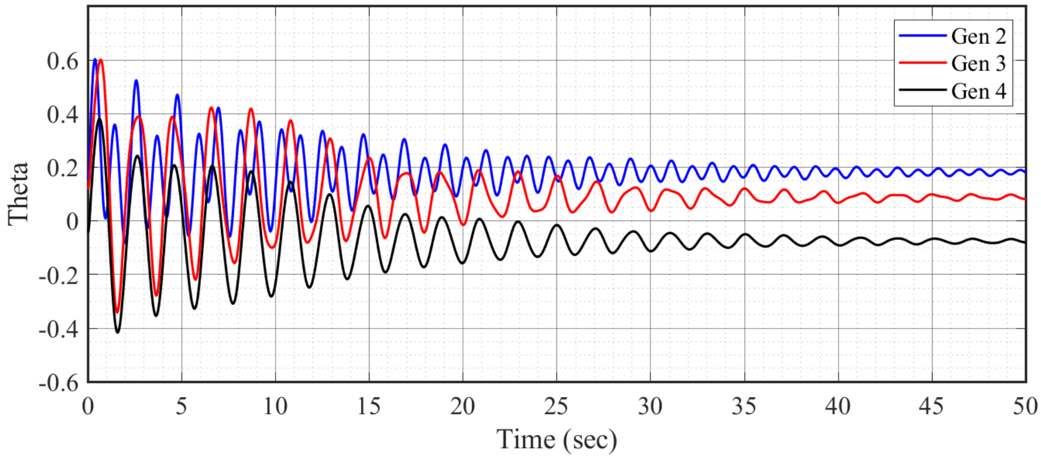

Figure 17: Generator angle vs time when Tc = 6 cycles

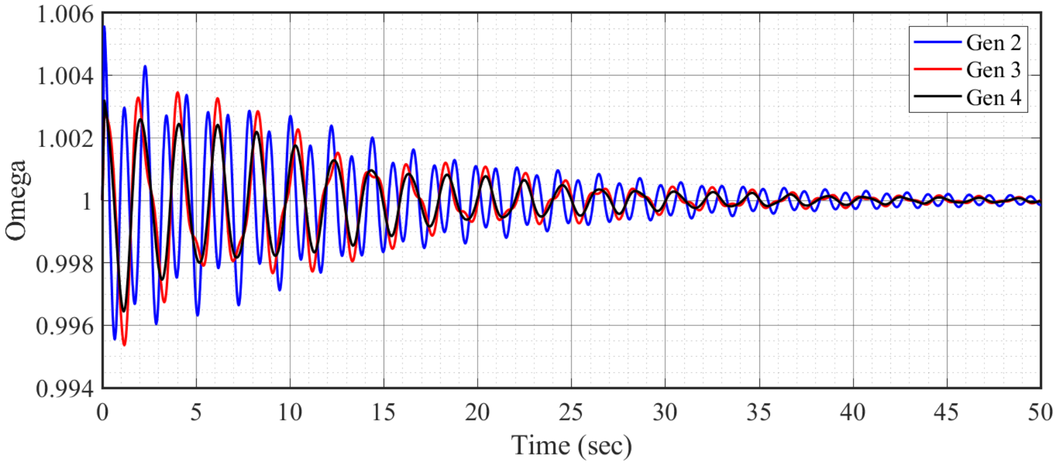

Figure 18: Generator speed vs time when Tc = 6 cycles

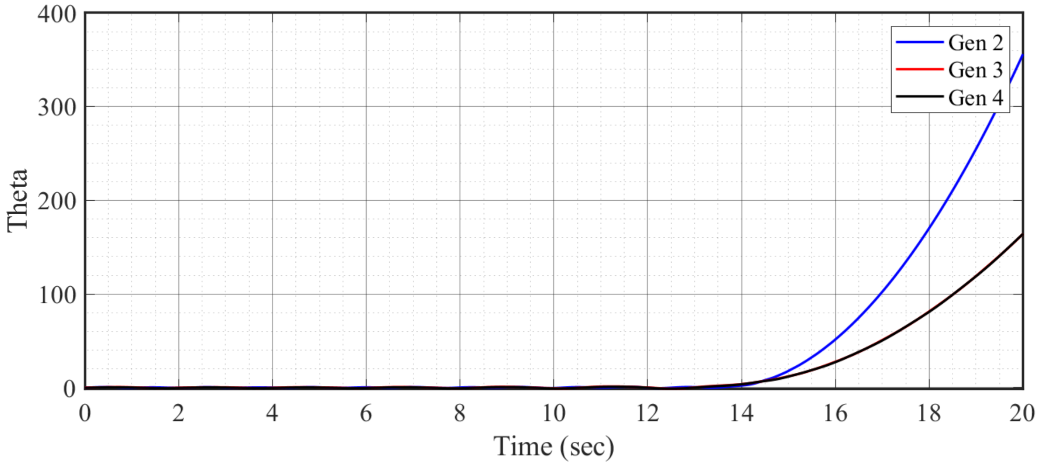

Figure 19: Generator angle vs time when Tc = 7 cycles

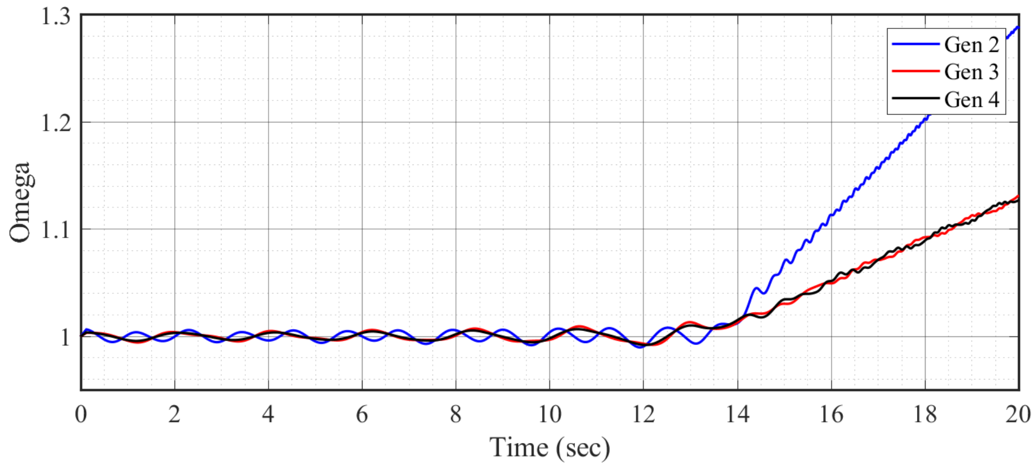

Figure 20: Generator speed vs time when Tc = 7 cycles

4.2 Type 2 Model

[htbp] The Type 2 model involves six state equations per generator, summing up to 18 state equations for the three generators. MATLAB's "vpasolve" function is employed to solve the system of equations, with inputs extracted directly from the .cdf files. The simulation procedure includes dynamic initialization, along with subsequent analyses for small signal stability and transient stability.

4.2.1 Dynamic Initialization

The system equations are initialized with typical initial values and subsequently solved to attain the dynamic initialization state. The resulting initialized condition is illustrated in table 4. The obtained results appear to be accurate, closely aligning with their typical values, with no deviations observed.

Steady state value	State
0.7301236253	θ_2
0.6568749844	θ_3
0.5478365253	θ_4
1.0	ω_2
1.0	ω_3
1.0	ω_4
0.9094668282	E_{q_2}
0.9671886236	E_{q_3}
0.8918971362	E_{q_4}
0.5906454497	E_{d_2}
0.5785839157	E_{d_3}
0.5996466689	E_{d_4}
1.7448315172	E_{fd_2}
1.8294497080	E_{fd_3}
1.7184269985	E_{fd_4}
7.0001442752	P_{m_2}
7.1897445896	P_{m_3}
6.9992763992	P_{m_4}

Table 4: Dynamic Initialization of Type 2 model

4.2.2 Small Signal Stability

The eigen values are calculated from dynamic initialization results and is tabulated in table 5. Evaluation of these eigenvalues confirms the system is small signal stable since all eigen values posses negative real part. However, low damping is observed for certain modes, indicating a requirement for PSS. Subsequently, PSS design is conducted and implemented across all three generators.

Mode	Eigen Values	Frequency	Damping	Mode Type
1	-0.42089 + 6.3554i	1.0115	6.61	Local
2	-0.42089 - 6.3554i	1.0115	6.61	Local
3	-0.28902 + 5.657i	0.9003	5.1	Intra
4	-0.28902 - 5.657i	0.9003	5.1	Intra
5	-6.8438 + 0i	0	100	
6	-6.3271 + 0i	0	100	
7	-4.4875 + 0i	0	100	
8	-0.011982 + 3.3021i	0.5255	0.36	Intra
9	-0.011982 - 3.3021i	0.5255	0.36	Intra
10	-0.13194 + 0i	0	100	
11	-0.10983 + 0i	0	100	
12	-0.0173 + 0i	0	100	
13	-0.0033303 + 0i	0	100	
14	-0.0033331 + 0i	0	100	
15	-0.0033332 + 0i	0	100	
16	-100 + 0i	0	100	
17	-100 + 0i	0	100	
18	-100 + 0i	0	100	

Table 5: Eigen values of Type 2 model

4.2.3 Type 2 with PSS

Considering the observed damping in the Type 2 system falls below 5%, it is imperative to integrate Power System Stabilizers (PSS) to enhance the system's small signal response. PSS has been devised for all three generators, followed by small signal analysis. The PSS parameters have been fine-tuned through trial and error, as detailed in Table 7. Post PSS implementation, a subsequent small signal analysis was conducted, with the resulting eigenvalues tabulated in Table 8. The outcomes illustrate an enhancement in system

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7	Mode 8	Mode 9
θ_2	1	0.0001	0	0.1173	0.1173	0.0001	0.0007	0	0.9998
θ_3	0.0009	0	0	0.9999	0.9999	0.0009	0	0.0005	0.0854
θ_4	0.0006	0	0	0.5942	0.5942	0.0006	0	0.0009	0.0043
ω_2	0	0	0	0.1173	0.1173	0	0.0008	0	1
ω_3	0	0	0	1	1	0.0003	0	0.0005	0.0854
ω_4	0	0	0	0.5942	0.5942	0.0002	0	0.001	0.0043
E_{q_2}	0	0	0	0.0168	0.0168	0	0.0046	0	0.061
E_{q_3}	0	0	0	0.04	0.04	0.0001	0.0001	0.0048	0.0057
E_{q_4}	0	0	0	0.0521	0.0521	0.0001	0	0.0071	0.0001
E_{d_2}	0	0	0	0.003	0.003	0.0289	1	0.006	0.038
E_{d_3}	0.0002	0	0	0.0594	0.0594	1	0.0175	0.8252	0.0028
E_{d_4}	0.0002	0	0	0.05	0.05	0.8012	0.0018	1	0.0001
E_{fd_2}	0	0	0	0	0	0	0	0	0
E_{fd_3}	0	0	0	0	0	0	0	0	0
E_{fd_4}	0	0	0	0	0	0	0	0	0
P_{m_2}	0.0747	1	0.0027	0	0	0.0001	0	0	0
P_{m_3}	1	0.0401	0.6291	0.0001	0.0001	0.0011	0	0	0
P_{m_4}	0.6609	0.0067	1	0	0	0.0007	0	0	0

	Mode 10	Mode 11	Mode 12	Mode 13	Mode 14	Mode 15	Mode 16	Mode 17	Mode 18
θ_2	0.9998	0.0219	0.0219	0.0061	0.0274	0.0042	0	0	0
θ_3	0.0854	0.5562	0.5562	0.0186	0.0033	0.0338	0	0	0
θ_4	0.0043	0.9998	0.9998	0.0312	0.0002	0.0322	0	0	0
ω_2	1	0.0219	0.0219	0.0061	0.0275	0.0042	0	0	0
ω_3	0.0854	0.5563	0.5563	0.0187	0.0033	0.0339	0	0	0
ω_4	0.0043	1	1	0.0313	0.0002	0.0323	0	0	0
E_{q_2}	0.061	0.0012	0.0012	0.0721	1	0.0731	0	0	0
E_{q_3}	0.0057	0.0459	0.0459	1	0.1051	0.5387	0	0	0
E_{q_4}	0.0001	0.1025	0.1025	0.6543	0.0012	1	0	0	0
E_{d_2}	0.038	0.0011	0.0011	0.0007	0.0008	0	0	0	0
E_{d_3}	0.0028	0.0206	0.0206	0.0019	0.0001	0.0005	0	0	0
E_{d_4}	0.0001	0.0327	0.0327	0.0024	0	0.0015	0	0	0
E_{fd_2}	0	0	0	0	0	0	1	0	0
E_{fd_4}	0	0	0	0	0	0	0	1	0
P_{m_2}	0	0	0	0	0	0	0	0	1
P_{m_3}	0	0	0	0	0	0	0	0	0
P_{m_4}	0	0	0	0	0	0	0	0	0

Table 6: Participation Matrix for Type 2 model

	T_w	T_a	T_b	K_s
G2	100	3	200	0.5
G3	100	3	200	0.5
G4	100	3	200	0.5

Table 7: PSS parameters

damping, ensuring it remains above 5%. Nevertheless, the system's mode count has expanded to 24, owing to the introduction of 2 modes per PSS.

Mode	Eigen Values	Frequency	Damping	Mode Type
1	-0.001 + 0i	0	100	
2	-0.001 + 0i	0	100	
3	-0.001 + 0i	0	100	
4	-0.1 + 0i	0	100	
5	-0.1 + 0i	0	100	
6	-0.1 + 0i	0	100	
7	-0.60016 + 6.3558i	1.0116	9.4009	Local
8	-0.60016 - 6.3558i	1.0116	9.4009	Local
9	-0.45728 + 5.6572i	0.90037	8.0568	Intra
10	-0.45728 - 5.6572i	0.90037	8.0568	Intra
11	-6.827 + 0i	0	100	
12	-6.3172 + 0i	0	100	
13	-4.4984 + 0i	0	100	
14	-0.17696 + 3.3046i	0.52594	5.3471	Intra
15	-0.17696 - 3.3046i	0.52594	5.3471	Intra
16	-0.13165 + 0i	0	100	
17	-0.10956 + 0i	0	100	
18	-0.017183 + 0i	0	100	
19	-0.0033303 + 0i	0	100	
20	-0.0033331 + 0i	0	100	
21	-0.0033332 + 0i	0	100	
22	-100 + 0i	0	100	
23	-100 + 0i	0	100	
24	-100 + 0i	0	100	

Table 8: Eigen values of Type 2 model with PSS

To compare the improvement in system response, eigen values are plot in fig 21. There has been some improvement in damping has been noted for frequency modes.

4.2.4 Transient Stability

[htbp] The integration of PSS into the Type 2 system introduces a total of 24 state variables, in addition to the existing 6 intrinsic variables. However, employing Euler integration for analyzing this extended system presents stability issues, impeding successful convergence during transient analysis. To address this challenge, adopting more advanced integration methods such as Runge-Kutta 4 or other sophisticated techniques becomes im-

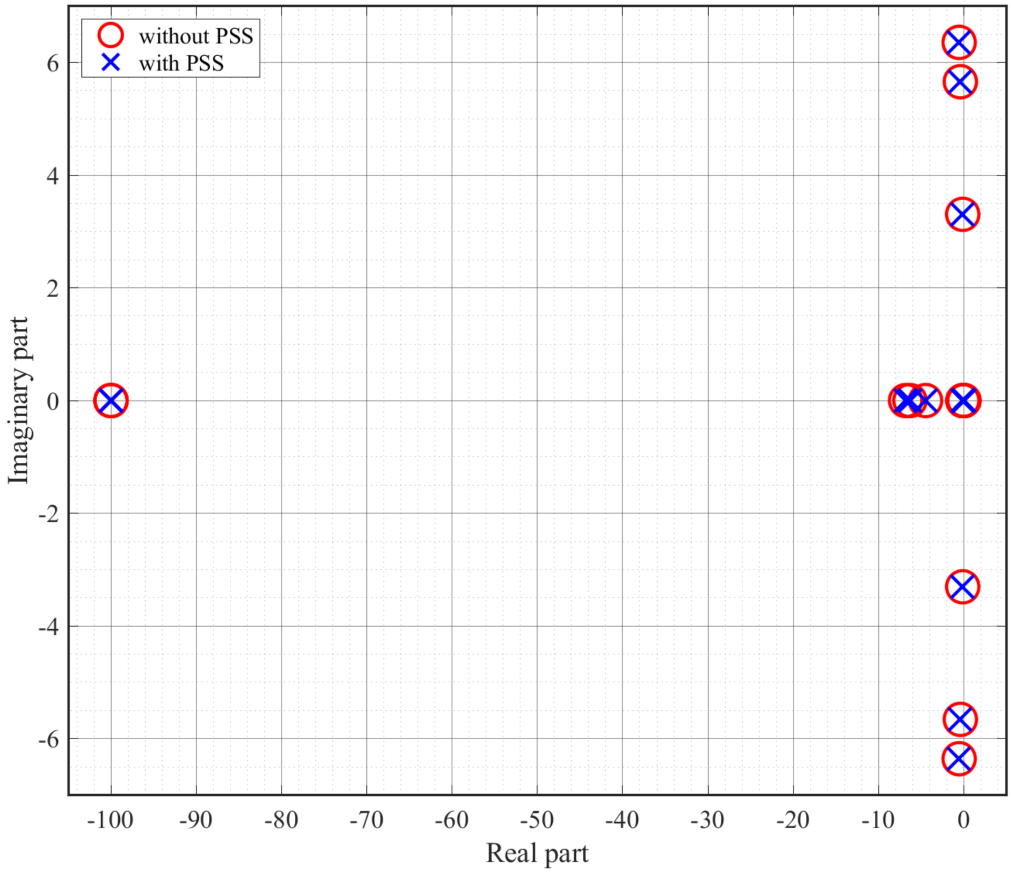

Figure 21: Eigen values of type 2 system with and without PSS

perative. These approaches offer enhanced stability and accuracy, making them better suited for handling the complexities inherent in large-scale power system simulations.

5 Conclusion

The small signal and transient stability analysis with Kundur two area system is implemented in MATLAB for Type 2 and Type 3 models. Following the dynamic initialization of these models, small signal analysis was undertaken to access system stability. This analysis revealed that both Type 2 and Type 3 models exhibited stable behavior, with modes categorized into inter, intra, and local modes.

Furthermore, a deeper investigation into mode behavior was conducted through the calculation of participation matrices, shedding light on the extent of state involvement in each mode. Notably, it was observed that certain modes displayed insufficient damping ratios, necessitating the implementation of Power System Stabilizers (PSS) within the Type 2 model. The subsequent integration of PSS yielded promising results, effectively enhancing system damping.

Additionally, transient analysis was performed to assess the system's response to disturbances. Through this analysis, a critical clearing angle of 6 cycles was identified for the Type 3 model, providing valuable insights into system resilience during transient events. Overall, these findings underscore the importance of comprehensive stability analysis in ensuring the robustness of power systems.

References

[1] P. Kundur, "Power system stability," *Power system stability and control*, vol. 10, pp. 7–1, 2007.

[2] D. J. Higham and N. J. Higham, *MATLAB guide*. SIAM, 2016.

Appendix

main.m

```
%% Power Flow
clc
clear all; close all;
cd("C:\Users\athul.p\Documents\GitHub\Power_System_Dynamics\HW6")
% cd("C:\Users\athul.p\Documents\GitHub\PowerSystemDynamic")
% Initializing Kundur 2 area system and importing data
n_bus = 11;
bus_data = importdata('ieee11bus.txt').data;
% bus_data = importdata('ieee11bus_allPV.txt').data;
branch_data = importdata('ieee11branch.txt').data;

% Ybus formation
t = 0; % 0 for without tap, 1 for with tap
Y = y_bus_calc(n_bus,bus_data,branch_data,t);

% Scheduled power calculation
base_MVA = 100;
P_inj = (bus_data(:,8) - bus_data(:,6)) / base_MVA;
Q_inj = (bus_data(:,9) - bus_data(:,7)) / base_MVA;

% Finding bus types
pv_i = find(bus_data(:,3) == 2);
pq_i = find(bus_data(:,3) == 0);
n_pv = length(pv_i);
n_pq = length(pq_i);

% Initializing Voltage magnitude and angles
V = bus_data(:,11);
V(V(:)==0) = 1;
T = zeros(n_bus,1);

% Newton Raphson Method
[V1_data,T1_data,T1] = NR(bus_data,V,T,P_inj,Q_inj,n_bus,Y,n_pq,pq_i);
% V = V1_data(:,end)
% T = T1_data(:,end)

% Fast Decoupled Mehod
%[V2_data,T2_data,T2] = FD(bus_data,V,T,P_inj,Q_inj,n_bus,Y,n_pq,pq_i);

% P,Q calculation after convergence
[P,Q] = PQ_calc(V1_data(:,size(V1_data,2)),T1_data(:,size(T1_data,2)),Y)
    ;
V = V1_data(:,size(V1_data,2));
T = T1_data(:,size(T1_data,2));
% [P,Q] = PQ_calc(V,T,Y)

```

```
% plotting convergence curves
% mplot([1:size(V1_data,2)],T1,[1:size(V1_data,2)],T1)

bus_data = importdata('ieee11bus_ynet.txt').data;
branch_data = importdata('ieee11branch_ynet.txt').data;
t = 0; % 0 for without tap, 1 for with tap
n_bus = 14;
Y = y_bus_calc(n_bus,bus_data,branch_data,t);

% calculating Ygen matrix

Y_gen = Y_gen_calc(Y)

P_gen = P(2:4);
Q_gen = Q(2:4);
V(12:14) = V(2:4);
T(12:14) = T(2:4);

% calculating Igen
I_gen = (P_gen - i*Q_gen)./(V(12:14).*exp(-i*T(12:14)));

X_gen = (((0.3+0.55)/2)*base_MVA/900);

% calculating Egen
E = (V(12:14).*exp(i*T(12:14))) + (i*X_gen.*I_gen);
[Theta, E_g] = cart2pol(real(E),imag(E))
E_g = [V(1); E_g]
Theta = [T(1); Theta]

%% Dynamic Initialization (Type 3)
t = sym('t', [1 3]);
w = sym('w', [1 3]);

F = type3(t, w, P_gen, Y_gen, E_g)
solVal0 = [0, 0, 0, 1, 1, 1]
vars = [t, w];
vpa(F,3)

sol = vpasolve(F == 0, vars, solVal0)

%% Small Signal Stability (Type 3)
trans_init = [];
% making Jacobian
for i = 1:length(vars)
    sol_val{i} = sol.(char(vars(i)));
    trans_init = [trans_init double(sol.(char(vars(i))))];
end
J = jacobian(F, vars);

% substituting solution
J_val = subs(J, vars, sol_val)
```

```
J_val = double(J_val);

% eigen value and vectors
[r_ev, eig_val] = eig(J_val);
l_ev = inv(r_ev);

% participation factor
norm_p = [];
for i = 1: length(vars)
    p_mat = r_ev(:,i)*l_ev(i,:);
    diag_vec = abs(diag(p_mat));
    norm_p = [norm_p diag_vec./max(diag_vec)];
end

% calculating mode frequency
eigen_frequency_mode= abs(imag(diag(eig_val)))/2/pi;
eigen = diag(eig_val);
100*(-real(eigen)./abs(eigen))

%% Transient Stability (Type 3)
% Ybus
Y_pre = Y;
Y_fault = y_fault_update();
Y_post = y_post_update();

% generating fault on system equations
Y_gen_fault = Y_gen_calc(Y_fault);
F_fault = type3(t, w, P_gen, Y_gen_fault, E_g);

% generating post fault system equations
Y_gen_post = Y_gen_calc(Y_post);
F_post = type3(t, w, P_gen, Y_gen_post, E_g);

% simulation parameters
h = 1e-3; Tf = 0; Tc = 7/60; Ts = 20;

% prefault scenario
x_init = trans_init;
x_step = [];
time_step = [];
[x_step, time_step] = Euler_step(h,x_init,F,vars,0,Tf-h);
x_data = [x_init' x_step];
time_data = [0 time_step];

% fault on scenario
x_init = x_data(:,end)';
[x_step, time_step] = Euler_step(h,x_init,F_fault,vars,Tf,Tf+Tc);
x_data = [x_data x_step];
time_data = [time_data time_step];

% post fault scenario
```

```
x_init = x_data(:,end)';
[x_step, time_step] = Euler_step(h,x_init,F_post,vars,Tf+Tc+h,Ts);
x_data = [x_data x_step];
time_data = [time_data time_step];

% seperating theta and omega variables
t_data = x_data(1:3,:);
w_data = x_data(4:6,:);

% plotting the graph
mplot(time_data, t_data, 'Theta')
mplot(time_data, w_data, 'Omega')

%% Dynamic Initialization (Type 2)
t = sym('t', [1 3]);
w = sym('w', [1 3]);
Eq = sym('Eq', [1 3]);
Ed = sym('Ed', [1 3]);
Efd = sym('Efd', [1 3]);
Pm = sym('Pm', [1 3]);
Vref = sym('Vref', [1 3]);
Pc = sym('Pc', [1 3]);
Vw = sym('Vw', [1 3]);
Vs = sym('Vs', [1 3]);

vars = [t, w, Eq, Ed, Efd, Pm, Vref, Pc, Vw, Vs];
F = type2(t, w, Eq, Ed, Efd, Pm, Vref, Pc, Vw, Vs, P_gen, Y_gen, E_g, V,
    T);

x0 = [0.685 0.614 0.504  1 1 1  0.935 0.991 0.916  0.549 0.537 0.559
   1.845 1.935 1.813  7.013 7.204 7.013 1.019 1.040 1.019 7.013 7.204
   7.013 1 1 1 1 1 1];
sol = vpasolve(F,vars,x0)

Small Signal Stability (Type 2)
state_vars = [t, w, Eq, Ed, Efd, Pm];

% making Jacobian
J = jacobian(F(1:18), state_vars);
sol_val = {};
for i = 1:length(state_vars)
    sol_val{i} = sol.(char(state_vars(i)));
end

% substituting solution
J_val = subs(J, state_vars, sol_val);
J_val = double(J_val);

% eigen value and vectors
[r_ev, eig_val] = eig(J_val);
l_ev = inv(r_ev);
```

```

% participation factor
norm_p = [];
for i = 1: length(state_vars)
    p_mat = r_ev(:,i)*l_ev(i,:);
    diag_vec = abs(diag(p_mat));
    norm_p = [norm_p diag_vec./max(diag_vec)];
end

% calculating mode frequency
eigen = diag(eig_val);
eigen_frequency_mode= abs(imag(eigen))/2/pi;
damping = 100*(-real(eigen)./abs(eigen));
figure()
plot(real(eigen),imag(eigen),"x")

%% PSS Analysis

pss_vars = [t, w, Eq, Ed, Efd, Pm, Vw, Vs];
F = type2_pss(t, w, Eq, Ed, Efd, Pm, Vref, Pc, Vw, Vs, P_gen, Y_gen, E_g
    , V, T);

sol = vpasolve(F,vars,x0)
F_pss = [F(1:18);F(25:30)];

% making Jacobian
J = jacobian(F_pss, pss_vars);
sol_val = {};
for i = 1:length(pss_vars)
    sol_val{i} = sol.(char(pss_vars(i)));
end

% substituting solution
J_val = subs(J, pss_vars, sol_val);
J_val = double(J_val);

[r_ev, eig_val] = eig(J_val);
eigen_pss = diag(eig_val);
eigen_frequency_mode_pss= abs(imag(eigen_pss))/2/pi;
damping_pss = 100*(-real(eigen_pss)./abs(eigen_pss))

figure()

plot(real(eigen),imag(eigen),"x")
hold on
plot(real(eigen_pss),imag(eigen_pss),"o")
mplot_eigen(real(eigen),imag(eigen),real(eigen_pss),imag(eigen_pss))

%% Transient Stability (Type 2)
F = type2(t, w, Eq, Ed, Efd, Pm, Vref, Pc, Vw, Vs, P_gen, Y_gen, E_g, V,
    T);
```

```

x0 = [0.685 0.614 0.504  1 1 1  0.935 0.991 0.916  0.549 0.537 0.559
    1.845 1.935 1.813  7.013 7.204 7.013 1.019 1.040 1.019 7.013 7.204
    7.013 1 1 1 1 1 1];
sol = vpasolve(F,vars,x0)

trans_init = [];
for i = 1:length(vars)
    trans_init = [trans_init double(sol.(char(vars(i))))];
end

% generating fault on system equations
Y_gen_fault = Y_gen_calc(Y_fault);
F_fault = type2(t, w, Eq, Ed, Efd, Pm, Vref, Pc, Vw, Vs, P_gen,
    Y_gen_fault, E_g, V, T);

% generating post fault system equations
Y_gen_post = Y_gen_calc(Y_post);
F_post = type2(t, w, Eq, Ed, Efd, Pm, Vref, Pc, Vw, Vs, P_gen,
    Y_gen_post, E_g, V, T);

% simulation parameters
h = 1e-3; Tf = 0; Tc = 6/60; Ts = 10;

x_step = [];
time_step = [];
% prefault scenario
x_init = trans_init;
[x_step, time_step] = Euler_step(h,x_init,F,vars,0,Tf-h);
x_data = [x_init' x_step];
time_data = [0 time_step];

% fault on scenario
x_init = x_data(:,end)';
[x_step, time_step] = Euler_step(h,x_init,F_fault,vars,Tf,Tf+Tc);
x_data = [x_data x_step];
time_data = [time_data time_step];

% post fault scenario
x_init = x_data(:,end)';
[x_step, time_step] = Euler_step(h,x_init,F_post,vars,Tf+Tc+h,Ts);
x_data = [x_data x_step];
time_data = [time_data time_step];

% seperating theta and omega variables
t_data = x_data(1:3,:);
w_data = x_data(4:6,:);

% plotting the graph
mplot(time_data, t_data, 'Theta')
mplot(time_data, w_data, 'Omega')
```

dpdq_calc.m

```
function [del_P, del_Q] = dpdq_calc(bus_data,V,T,P_inj,Q_inj,n_bus,Y)
    P = zeros(n_bus,1);
    Q = zeros(n_bus,1);
    Pi = 1;
    Qi = 1;
    for i = 1:n_bus
        if(bus_data(i,3) ~= 3)
            for j = 1:n_bus
                P(i) = P(i) + V(i)*V(j)*abs(Y(i,j))*cos(T(i)-T(j)-angle(
    Y(i,j)));
                Q(i) = Q(i) + V(i)*V(j)*abs(Y(i,j))*sin(T(i)-T(j)-angle(
    Y(i,j)));
            end
            del_P(Pi) = P_inj(i) - P(i);
            Pi = Pi+1;
            if(bus_data(i,3) == 0)
                del_Q(Qi) = Q_inj(i) - Q(i);
                Qi = Qi+1;
            end
        end
    end
end
```

Euler_step.m

```
%% Euler Integration
function [x_data, time_data] = Euler_step(h,x_init,F,vars,T_start,T_stop
    )
    x_k = x_init;
    x_data = [];
    time_data = [];
    F_ev = matlabFunction(F,"Vars", vars);

    for time = T_start:h:T_stop
        % F_val1 = double(subs(F, vars, x_k));

        x_k_c = num2cell(x_k);
        F_val = feval(F_ev,x_k_c{:});
        x_kn = x_k' + (F_val*h);
        time_data = [time_data time];
        x_data = [x_data x_kn];
        x_k = x_kn';
        if(mod(time,1) == 0)
            fprintf("%d seconds of simulation finished \n",time);
        end
    end
end
```

FD.m

```
function [V_data,T_data,Tol_data] = FD(bus_data,V,T,P_inj,Q_inj,n_bus,Y,
    n_pq,pq_i)
    % Initializing index
    B = imag(Y);
    B_T =  - B(2:n_bus,2:n_bus);
    B_V = - B(pq_i,pq_i);
    i = 0;
    Tol = 1;
    del_T = zeros(n_bus,1);
    del_V = zeros(n_bus,1);

    % Iteration loop
    while(Tol > 1e-3 & i < 100)
        i = i+1;
        V = V+del_V;
        T = T+del_T;
        T_data(:,i) = T;
        V_data(:,i) = V;
        [del_P, del_Q] = dpdq_calc(bus_data,V,T,P_inj,Q_inj,n_bus,Y);
        P_T = del_P'./V(2:n_bus);
        d_T = fwd_bwd(B_T,P_T);
        Q_V = del_Q'./V(pq_i);
        d_V = fwd_bwd(B_V,Q_V);
        del_T = [0 d_T]'; % angle calculation
        for j = 1:n_pq
            del_V(pq_i(j)) = d_V(j); % magnitude calculation
        end
        Tol = max(abs([P_T; Q_V]));
        Tol_data(i) = Tol;
    end
end
```

fwd_bwd.m

```
function x = fwd_bwd(A,b)
    [L, U] = LU(A);

    % Forward Substitution
    for k = 1:length(A)
        s = 0;
        for j = 1:k-1
            s = s + (L(k,j)*y(j));
        end
        y(k) = (b(k) - s) / L(k,k);
    end

    % Backward Substitution
    for k = length(A):-1:1
        s = 0;
        for j = k+1:length(A)
            s = s + (U(k,j)*x(j));
```

```
        end
        x(k) = y(k) - s;
    end
end
```

J_calc.m

```
function J = J_calc(bus_data,V,T,Y,n_bus,n_pq,pq_i)
    % J1 calculation
    J1 = zeros(n_bus-1);
    for i = 1:n_bus
        for j = 1:n_bus
            if(bus_data(i,3) ~=3 & bus_data(j,3) ~=3)
                if(i==j)
                    for k = 1:n_bus
                        J1(i-1,j-1) = J1(i-1,j-1)+(V(i)*V(k)*abs(Y(i,k))*sin(angle(Y(i,k))-T(i)+T(k)));
                    end
                    J1(i-1,j-1) = J1(i-1,j-1) - ((V(i)^2) * (imag(Y(i,i))));
                else
                    J1(i-1,j-1) = -V(i)*V(j)*abs(Y(i,j))*sin(angle(Y(i,j))-T(i)+T(j));
                end
            end
        end
    end
    J1;

    % J2 calculation
    J2 = zeros(n_bus-1,n_pq);
    for i = 2:n_bus
        for j = 1:n_pq
            n = pq_i(j);
            if(n == i)
                for k = 1:n_bus
                    J2(i-1,j) = J2(i-1,j)+(V(i)*V(k)*abs(Y(i,k))*cos(angle(Y(i,k))-T(i)+T(k)));
                end
                J2(i-1,j) = (J2(i-1,j) + ((V(i)^2) * (real(Y(i,i)))))/V(i);
            else
                J2(i-1,j) = V(i)*abs(Y(i,n))*cos(angle(Y(i,n))-T(i)+T(n));
            end
        end
    end
    J2;

    % J3 calculation
    J3 = zeros(n_pq,n_bus-1);
```

```
    for i = 1:n_pq
        n = pq_i(i);
        for j = 2:n_bus
            if(n==j)
                for k = 1:n_bus
                    J3(i,j-1) = J3(i,j-1)+(V(n)*V(k)*abs(Y(n,k))*cos(
    angle(Y(n,k))-T(n)+T(k)));
                end
                J3(i,j-1) = J3(i,j-1) - ((V(n)^2) * (real(Y(n,n))));
            else
                J3(i,j-1) = -V(n)*V(j)*abs(Y(n,j))*cos(angle(Y(n,j))-T(n
    )+T(j));
            end
        end
    end
    J3;

    % J4 calculation
    J4 = zeros(n_pq);
    for i = 1:n_pq
        n1 = pq_i(i);
        for j = 1:n_pq
            n2 = pq_i(j);
            if(n1==n2)
                for k = 1:n_bus
                    J4(i,j) = J4(i,j)+(V(n1)*V(k)*abs(Y(n1,k))*sin(angle
    (Y(n1,k))-T(n1)+T(k)));
                end
                J4(i,j) = (- J4(i,j) - ((V(n1)^2) * (imag(Y(n1,n1)))))/V
    (n1);
            else
                J4(i,j) = -V(n1)*abs(Y(n1,n2))*sin(angle(Y(n1,n2))-T(n1)
    +T(n2));
            end
        end
    end
    J4;
    J = [J1, J2; J3, J4];
end
```

LU.m

```
function [L, U] = LU(a)
    Q = zeros(length(a));
    for j = 1:length(a)
        for k = j:length(a)
            s = 0;
            for m = 1:j-1
                s = s + (Q(k,m)*Q(m,j));
            end
            Q(k,j) = a(k,j) - s;
```

```
        end
        if j < length(a)
            for k = j+1:length(a)
                s = 0;
                for m = 1:j-1
                    s = s + (Q(j,m)*Q(m,k));
                end
                Q(j,k) = (a(j,k) - s) / Q(j,j);
            end
        end
    end
    L = tril(Q);
    U = Q - L + eye(length(a));
end
```

mplot.m

```
function mplot(x,y,label)
    x_label = 'Time (sec)'; % x axis label
    y_label = label; % y axis label

    legend_name = {'Gen 2','Gen 3', 'Gen 4'}; % legend names

    %%%%%%% Theta fig
    figure('Renderer', 'painters', 'Position', [10 10 1000 400])
    plot(x,y(1,:),'-b','LineWidth',1.5)
    hold on
    plot(x,y(2,:),'-r','LineWidth',1.5)
    plot(x,y(3,:),'-k','LineWidth',1.5)
    xlabel(x_label,'FontSize',18,'FontName','Times New Roman')
    ylabel(y_label,'FontSize',18,'FontName','Times New Roman')
    legend (legend_name,'Location','northeast')
    set(gca,'fontsize',16,'Fontname','Times New Roman','GridAlpha',0.5)
    ax = gca

    ax.XRuler.Axle.LineWidth = 1.5;
    ax.YRuler.Axle.LineWidth = 1.5;
    grid
    grid minor
    % legend (legend_name,'Location','southeast')
    saveas(gca,[label '_plot.png'])
end
```

mplot_eigen.m

```
function mplot_eigen(x1,y1,x2,y2)
    x_label = 'Real part'; % x axis label
    y_label = 'Imaginary part'; % y axis label

    legend_name = {'without PSS','with PSS'}; % legend names

```

```
    %%%%%%% Theta fig
    figure('Renderer', 'painters', 'Position', [10 10 1000 800])
    plot(x1,y1,'xb','LineWidth',1.5)
    hold on
    plot(x2,y2,'or','LineWidth',1.5)
    xlim([-105.0 5.0])
    ylim([-7.0 7.0])
    % plot(x,y(3,:),'-k','LineWidth',1.5)
    xlabel(x_label,'FontSize',18,'FontName','Times New Roman')
    ylabel(y_label,'FontSize',18,'FontName','Times New Roman')
    legend (legend_name,'Location','northeast')
    set(gca,'fontsize',16,'Fontname','Times New Roman','GridAlpha',0.5)
    ax = gca

    ax.XRuler.Axle.LineWidth = 1.5;
    ax.YRuler.Axle.LineWidth = 1.5;
    grid
    grid minor
    legend (legend_name,'Location','northwest')
    saveas(gca,['eigen_plot.png'])
end
```

NR.m

```
function [V_data,T_data,Tol_data] = NR(bus_data,V,T,P_inj,Q_inj,n_bus,Y,
    n_pq,pq_i)
    % Initializing index
    i = 0;
    Tol = 1;
    del_T = zeros(n_bus,1);
    del_V = zeros(n_bus,1);

    % Iteration loop
    while(Tol > 1e-3 & i < 100)
        i = i+1;
        V = V+del_V;
        T = T+del_T;
        T_data(:,i) = T;
        V_data(:,i) = V;
        [del_P, del_Q] = dpdq_calc(bus_data,V,T,P_inj,Q_inj,n_bus,Y);
        dpdq = [del_P, del_Q]; % mismatch calculation
        J = J_calc(bus_data,V,T,Y,n_bus,n_pq,pq_i); % Jacobian
    calculation
        delta = fwd_bwd(J,dpdq); % finding errors
        del_T = [0 delta(1:n_bus-1)]';
        for j = 1:n_pq
            del_V(pq_i(j)) = delta(n_bus+j-1);
        end
        Tol = max(abs(delta)); % updating error for convergence
        Tol_data(i) = Tol;
    end
```

```
end
```

PQ_calc.m

```
function [P,Q] = PQ_calc(V,T,Y)
    n_bus = size(V,1);
    P = zeros(n_bus,1);
    Q = zeros(n_bus,1);
    for i = 1:n_bus
        for j = 1:n_bus
            P(i) = P(i) + V(i)*V(j)*abs(Y(i,j))*cos(T(i)-T(j)-angle(Y(i,j)));
            Q(i) = Q(i) + V(i)*V(j)*abs(Y(i,j))*sin(T(i)-T(j)-angle(Y(i,j)));
        end
    end
end
```

type2.m

```
%%%%%%%%%%%%%%%%%% Type 2 Forumulation %%%%%%%%%%%%%%%%%%%%
function F = type2(t, w, Eq, Ed, Efd, Pm, Vref, Pc, Vw, Vs, P,Y,E, V,T)
    %Vref = [1.019 1.040 1.019];
    H = [6.5 6.175 6.175];
    Kd = 30;
    omega_s = 2*pi*60;
    Xd = (1.8 + 1.7)/2/9;
    Xdp = (0.3 + 0.55)/2/9;
    Ym = abs(Y);
    Ya = angle(Y);
    F = [];

    for k = 1:3
        Ep(k+1) = sqrt((Ed(k)*Ed(k)) + (Eq(k)*Eq(k)));
        g(k+1) = t(k)-(pi/2)+(atan(Eq(k)/Ed(k)));
    end
    Ep;
    for k = 1:3
        i = k+1;
        Pe = Ym(i,1)*Ep(i)*E(1)*cos(g(i)-Ya(i,1));
        Id = Ym(i,1)*E(1)*sin(t(k)-Ya(i,1));
        Iq = Ym(i,1)*E(1)*cos(t(k)-Ya(i,1));

        for j = 2:4
            Pe = Pe + Ym(i,j)*Ep(i)*Ep(j)*cos(g(i)-g(j)-Ya(i,j));
            Id = Id + Ym(i,j)*Ep(j)*sin(t(k)-g(j)-Ya(i,j));
            Iq = Iq + Ym(i,j)*Ep(j)*cos(t(k)-g(j)-Ya(i,j));
        end
        vpa(Id,3);
        vpa(Iq,3);
        % (w-1)ws; Pm - Pe - Kd(w-1)
```

```

        F1 = [(w(k)-1)*omega_s,
               (Pm(k)-Pe-(Kd*(w(k)-1)))/(2*H(k)*9),
               (-Eq(k)-((Xd-Xdp)*Id)+Efd(k))/8,
               (-Ed(k)+((Xd-Xdp)*Iq))/0.4,
               (-Efd(k)-(200*(Vref(k) - V(i))))/0.01,
               (-Pm(k)+Pc(k)+((1/0.05)*(1-w(k))))/300,
               Ed(k) - (V(i)*sin(t(k)-T(i))) + Iq*Xdp
               Eq(k) - (V(i)*cos(t(k)-T(i))) - Id*Xdp];

        F = [F F1];
    end
    F = [F(1,:) F(2,:) F(3,:) F(4,:) F(5,:) F(6,:) F(7,:) F(8,:)].';
    F = [F; Vw(1); Vw(2); Vw(3); Vs(1); Vs(2); Vs(3)];
end
```

type2_pss.m

```
%%%%%%%%%%%%%%%%%% Type 2 Forumulation %%%%%%%%%%%%%%%%%%%
function F = type2_pss(t, w, Eq, Ed, Efd, Pm, Vref, Pc, Vw, Vs, P,Y,E, V
    ,T)
    H = [6.5 6.175 6.175];
    Kd = 40;
    omega_s = 2*pi*60;
    Xd = (1.8 + 1.7)/2/9;
    Xdp = (0.3 + 0.55)/2/9;
    Ym = abs(Y);
    Ya = angle(Y);
    F = [];
    Tw = [1 1 1]*10;
    Ta = [1 1 1]*3;
    Tb = [1 1 1]*200;
    Ks = [1 1 1]*0.05;

    for k = 1:3
        Ep(k+1) = sqrt((Ed(k)*Ed(k)) + (Eq(k)*Eq(k)));
        g(k+1) = t(k)-(pi/2)+(atan(Eq(k)/Ed(k)));
    end
    Ep;
    for k = 1:3
        i = k+1;
        Pe = Ym(i,1)*Ep(i)*E(1)*cos(g(i)-Ya(i,1));
        Id = Ym(i,1)*E(1)*sin(t(k)-Ya(i,1));
        Iq = Ym(i,1)*E(1)*cos(t(k)-Ya(i,1));
        for j = 2:4
            Pe = Pe + Ym(i,j)*Ep(i)*Ep(j)*cos(g(i)-g(j)-Ya(i,j));
            Id = Id + Ym(i,j)*Ep(j)*sin(t(k)-g(j)-Ya(i,j));
            Iq = Iq + Ym(i,j)*Ep(j)*cos(t(k)-g(j)-Ya(i,j));
        end
        F1 = [(w(k)-1)*omega_s,
               (Pm(k)-Pe-(Kd*(w(k)-1)))/(2*H(k)*9),
```

```
                (-Eq(k)-((Xd-Xdp)*Id)+Efd(k))/8,
                (-Ed(k)+((Xd-Xdp)*Iq))/0.4,
                (-Efd(k)-(200*(Vref(k) - V(i))))/0.01,
                (-Pm(k)+Pc(k)+((1/0.05)*(1-w(k))))/300,
                Ed(k) - (V(i)*sin(t(k)-T(i))) + Iq*Xdp
                Eq(k) - (V(i)*cos(t(k)-T(i))) - Id*Xdp
                (-Vw(k) - Tw(k)*((w(k)-1)*omega_s))/Tw(k)
                (-Ks(k)*Vs(k)+Vw(k)+Ta(k)*((-Vw(k) - Tw(k)*((w(k)-1)*
    omega_s))/Tw(k)))/(Ks(k)*Tb(k))];
         F = [F F1];
     end
     F = [F(1,:) F(2,:) F(3,:) F(4,:) F(5,:) F(6,:) F(7,:) F(8,:) F(9,:)
    F(10,:)].';
end
```

type3.m

```
function F = type3(t,w,P,Y,E)
    H = [6.5 6.175 6.175];
    Kd = 2;
    omega_s = 2*pi*60;
    Ym = abs(Y);
    Ya = angle(Y);
    F = [];
    for k = 1:3
        i = k+1;
        Pe = Ym(i,1)*E(i)*E(1)*cos(t(k)-Ya(i,1));
        for j = 2:4
            Pe = Pe + Ym(i,j)*E(i)*E(j)*cos(t(k)-t(j-1)-Ya(i,j));
        end
        % (w-1)ws; Pm - Pe - Kd(w-1)
        F1 = [(w(k)-1)*omega_s,
              (P(k)-Pe-(Kd*(w(k)-1)))/(2*H(k)*9)];
        F = [F F1];
    end
    F = [F(1,:) F(2,:)].';
end
```

y_bus_calc.m

```
function Y = y_bus_calc(N_bs,D_bs,D_br,t)
    Y = zeros(N_bs);
    % Calculating elements of Ybus
    for k = 1:size(D_br,1)
        Y(D_br(k,1),D_br(k,1)) = Y(D_br(k,1),D_br(k,1)) + 1/(D_br(k,7) +
    i*D_br(k,8)) + i*D_br(k,9)/2;
        Y(D_br(k,2),D_br(k,2)) = Y(D_br(k,2),D_br(k,2)) + 1/(D_br(k,7) +
    i*D_br(k,8)) + i*D_br(k,9)/2;
        Y(D_br(k,1),D_br(k,2)) = -1/(D_br(k,7) + i*D_br(k,8));
        Y(D_br(k,2),D_br(k,1)) = Y(D_br(k,1),D_br(k,2));
    end
```

```
    for k = 1:N_bs
        Y(k,k) = Y(k,k) + D_bs(k,14) + i*D_bs(k,15);
    end

    % adjusting for taps
    if(t == 1)
        for k = 1:size(D_br,1)
            if(D_br(k,15) ~= 0)
                t = D_br(k,15)
                ((t^2) / i*D_br(k,8));
                Y(D_br(k,1),D_br(k,1)) = Y(D_br(k,1),D_br(k,1)) + Y(D_br(k,1),D_br(k,2)) - (Y(D_br(k,1),D_br(k,2)))/(t^2);
                Y(D_br(k,1),D_br(k,1));
                Y(D_br(k,1),D_br(k,2)) = Y(D_br(k,1),D_br(k,2))/t;
                Y(D_br(k,2),D_br(k,1)) = Y(D_br(k,1),D_br(k,2));
            end
        end
    end
end
```

y_fault_update.m

```
function Y =  y_fault_update(Y)
    bus_data = importdata('ieee11bus_ynet_fault.txt').data;
    branch_data = importdata('ieee11branch_ynet_fault.txt').data;
    t = 0; % 0 for without tap, 1 for with tap
    n_bus = 14;
    Y = y_bus_calc(n_bus,bus_data,branch_data,t);
end
```

y_post_update.m

```
function Y =  y_post_update()
    bus_data = importdata('ieee11bus_ynet_postfault.txt').data;
    branch_data = importdata('ieee11branch_ynet_postfault.txt').data;
    t = 0; % 0 for without tap, 1 for with tap
    n_bus = 14;
    Y = y_bus_calc(n_bus,bus_data,branch_data,t);
end
```

Y_gen_calc.m

```
function Y_gen = Y_gen_calc(Y)
    Y_gen = Y(1:4,1:4)-(Y(1:4,5:14)*inv(Y(5:14,5:14)) *Y(5:14,1:4));
end
```