Spring 2025

46705

Power Grid Analysis

Assignment 2

Subject: Fault Analysis & PowerFactory verification using automation

Submission date: 10 May 2025

Introduction

This assignment aims to practice the Learning Objectives of the Module on "Fault Analysis" of the 46705 "Power Grid Analysis" course.

- Calculate short-circuit currents of symmetrical faults in small grids.
- Apply symmetrical components and sequence networks in small grids and analyze unsymmetrical (single line-to-ground, line-to-line and double line-to-ground) faults.
- Calculate symmetrical or unsymmetrical fault currents using sequence bus impedance matrices in large grids.
- Describe the basic principles of distance relays, design their zones of protection in a distance protection scheme and determine the operating time for a given fault current from provided relay characteristics.

The assignment comprises five parts,

Part A requests you to explain and perform symmetrical and unsymmetrical fault analysis for small grids without the use of the computer program

Part B requests you to develop a Python program based on the previous load flow project that is capable of performing fault analysis of larger systems using the Zbus method for balanced and unbalanced faults. The program will be used to calculate short-circuit currents in the Nordic 32 system from the previous assignment and the results will be validated with PowerFactory

Part C requests that you determine zones of protection, determine appropriate distance relay settings and calculate the loadability of the relay.

Part D requests you identify the fault current quantities relating to the IEC 60909:0-2016 standard (denoted 60909 from now on) with PowerFactory, as well as automate the switching transient assessment through Python coding

Part E requests that you verify the results of your power flow and security assessment of the Nordic 32-Bus system from the previous project with PowerFactory and develop a Python script for contingency assessment to automate the process

Part A

Question A.1

Please explain why the zero sequence network of the transformer is different for different connections in Figure 1 for the second and third cases.

How would the circuits change if the Y-connected side was grounded with an impedance Zn?

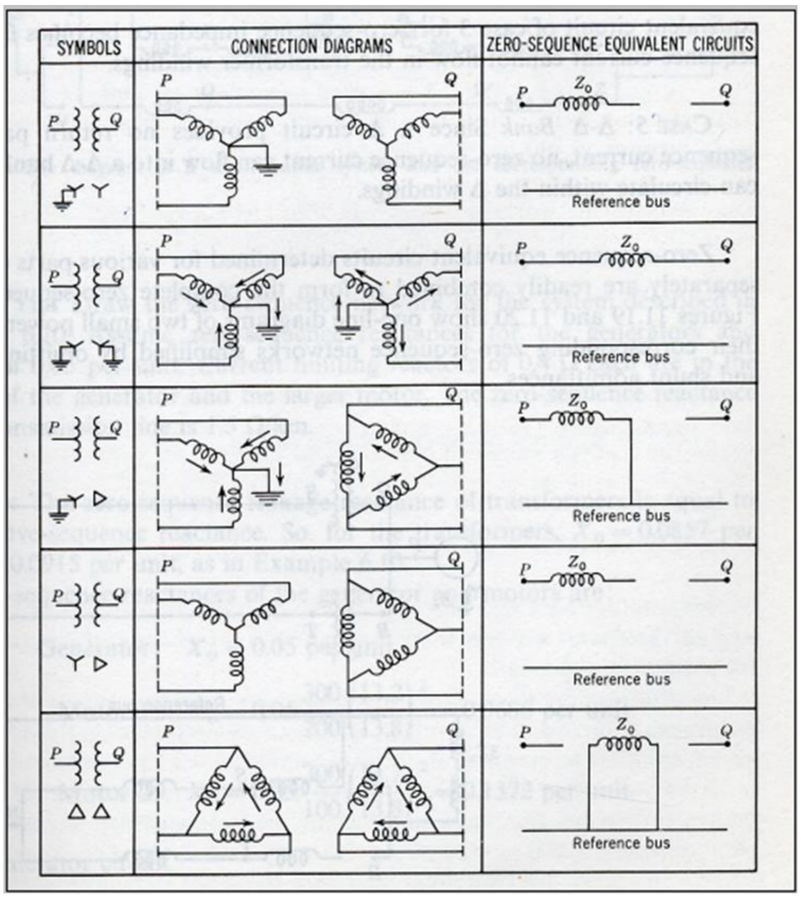


Figure 1. Zero-sequence equivalent circuits of transformers.

Question A.2

The single-line diagram of a three-phase power system is shown in Figure 2,

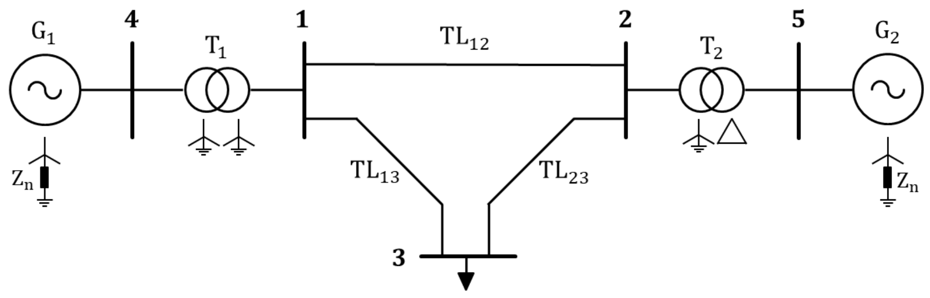


Figure 2. Grid topology of Question A.3

Equipment ratings and the per-unit sequence reactances are given as follows:

Synchronous Generators:

G1: 100 MVA, 25 kV, X1 = X2 = 0.2 pu, X0 = 0.05 pu, Xn = 0.03 pu

G2: 100 MVA, 13.8 kV, X1 = X2 = 0.2 pu, X0 = 0.05 pu, Xn = 0.03 pu

Transformers:

T1: 100 MVA, 25/230kV, X1 = X2 = 0.05 pu, X0 = 0.05 pu

T2: 100 MVA, 13.8/230kV, X1 = X2 = 0.05 pu, X0 = 0.05 pu

Transmission Lines:

TL12: 100 MVA, 230kV, X1 = X2 = 0.1 pu, X0 = 0.3 pu

TL13: 100 MVA, 230kV, X1 = X2 = 0.1 pu, X0 = 0.3 pu

TL23: 100 MVA, 230kV, X1 = X2 = 0.1 pu, X0 = 0.3 pu

Pre-fault voltage: 1.0 per unit, Neglect $\Delta - Y$ phase shifts.

- Draw the per-unit sequence networks for the above system
- How will the transformer's per-unit value change if the rating is changed to 200 MVA?
- Reduce the sequence networks to their Thévenin equivalents “looking in” at bus 3, using inspection of the sequence networks.

Question A.3

Compute the fault currents and voltages both in sequence and phase domain at the fault for the following faults at **bus 3**:

- A bolted three-phase fault.
- A bolted single line-to-ground fault.
- A bolted line-to-line fault.
- A bolted double line-to-ground fault.

Please include a figure with the connections of the sequence networks for unbalanced faults.

Part B

This part develops the algorithm to perform fault analysis verification (symmetrical + unsymmetrical).

The aim here is to get familiar with the main steps associated with the process of fault analysis. You will need to develop a small program in Python based on the load flow project functions, with the aim to calculate the static short-circuit currents and voltages for the Nordic32 Bus system.

The main script for the fault analysis program is provided in Table 1. You will find this script in DTU-Learn as *main4FA.py* under Assignment 2. The lines in the script, which are colored **RED** denote functions and **.py* files that you should write to complete the assignment. In BOLD, you will find the input data for the algorithm. That is, the location of the text file providing the network data (**filename**), location of the fault (**FaultBus**), type of the fault (**FaultType**), impedance of the fault (**FaultImpedance**), and the prefault voltage (**PrefaultVoltage**). *LoadNetworkData4FA* is an extension of the previously used *LoadNetworkData,* where the additional system parameters required for the calculation of short-circuit currents, e.g. *Zbus2,* are determined.

Table 1. A python script (main4FA.py) that carries out fault analysis

```
import LoadNetworkData4FA as lnd4fa # load the network data to global variables
filename = "./TestSystem4FA.txt"
lnd4fa.LoadNetworkData4FA(filename) # makes Zbus0 available as lndfa.Zbus0 etc.
# Carry out the fault analysis ...
FaultBus = 3
# FaultType:     0 = 3-phase balanced fault; 1 = Single Line-to-Ground fault;
#                2 = Line-to-Line fault; 3 = Double Line-to-Ground fault.
FaultType = 1
FaultImpedance = 0 # (in pu)
PrefaultVoltage = 1.000 # (in pu)
# Iph: phase current array (0: phase a; 1: phase b; 2: phase c).
# Vph_mat: phase line-to-ground voltages (rows: busses; columns: phases a, b, c).
Iph,Vph_mat = fa.FaultAnalysis(lnd4fa.Zbus0,lnd4fa.Zbus1,lnd4fa.Zbus2,lnd4fa.bus_to_ind,
                               FaultBus,FaultType,FaultImpedance,PrefaultVoltage)
# Display results
fa.DisplayFaultAnalysisResults(Iph,Vph_mat,FaultBus,FaultType,FaultImpedance,PrefaultVoltage)
print('**********End of Fault Analysis**********')
```

The functions that you have to write are:

- The function ***LoadNetworkData4FA*** (stored in ***LoadNetworkData4FA.py***) reads all the necessary information about the network (buses, generators, lines and transformers). The function creates the matrices and arrays needed to carry out fault analysis.
- The function ***FaultAnalysis()*** (stored in ***FaultAnalysis_46705.py***) is essential to the program. This function carries out the fault analysis calculations.
- The function ***DisplayFaultAnalysisResults()*** (stored in ***FaultAnalysis_46705.py***) displays the fault analysis results.

In the following, you will find guidelines for the construction of each of the above functions.

Task 1 - Construction of the ***FaultAnalysis()*** function

On DTU-LEARN, under the link for Assignment 2 you find the file FaultAnalysis_46705.py, which contains empty definitions of functions needed for the fault analysis computations (with hints). Your task in the following is to complete all of the functions in that file.

A Python skeleton for the *FaultAnalysis()* function is provided in Table 2.

The input to the function *FaultAnalysis()* is the N×N bus impedance matrices of the zero-, positive-, and negative-sequence networks, i.e., *Zbus0*, *Zbus1*, and *Zbus2*, respectively, a mapping from bus numbers to the corresponding indices in the bus matrices and arrays *bus_to_ind*, and the fault characteristics, i.e., the location of the fault (*fault_bus*), type of the fault (*fault_type*), impedance of the fault (*Zf*), and the prefault voltage (*Vf*).

As output, the function returns the following:

- the phase (fault) currents, *Iph*, an array that stores phase a, b, and c currents, i.e., element *Iph[0]* contains the phase a current, *Iph[1]* contains the phase b current, etc.,
- the phase (fault) line-to-ground voltages, *Vph_mat*, an Nx3 matrix that stores the voltages at each bus in rows and phase a, b, c in columns, i.e., element *Vph_mat[0][0]* contains voltage at bus 1 (first row), phase a (first column), element *Vph_mat[1][1]* contains voltage at bus 2 (second row), phase b (second column), element *Vph_mat[1][2]* contains voltage at bus 2 (second row), phase c (third column), etc.

The *FaultAnalysis()* function presented in Table 2, includes 4 sub-functions colored **RED**, which you are asked to write.

Table 2. Skeleton for the function FaultAnalysis()

```
def FaultAnalysis(Zbus0,Zbus1,Zbus2,bus_to_ind,fault_bus,fault_type,Zf,Vf):
    # calculate sequence fault currents
    Iseq = Calculate_Sequence_Fault_Currents(Zbus0,Zbus1,Zbus2,bus_to_ind,fault_bus,fault_type,Zf,Vf)
    # calculate sequence fault voltages
    Vseq_mat = Calculate_Sequence_Fault_Voltages(Zbus0,Zbus1,Zbus2,bus_to_ind,fault_bus,Vf,Iseq)
    # convert sequence currents to phase (fault) currents
    Iph = Convert_Sequence2Phase_Currents(Iseq)
    # convert sequence voltages to phase line-to-ground (fault) voltages
    Vph_mat = Convert_Sequence2Phase_Voltages(Vseq_mat)
    return Iph, Vph_mat
```

Question B.1

- Write the function *Calculate_Sequence_Fault_Currents()*, which calculates the sequence fault currents and stores them in the array *Iseq*, where element *Iseq[0]* contains the zero-sequence current, *Iph[1]* contains the positive-sequence current.
- Write the function *Calculate_Sequence_Fault_Voltages()*, which calculates the sequence (fault) voltages and stores them in the Nx3 matrix *Vseq_mat*, where element *Vseq_mat[0][0]* contains the bus 1 (first row) zero-sequence (first column) voltage, *Vseq[1][1]* contains the bus 2 (second row), positive-sequence (second column) voltage, *Vseq[1][2]* contains the bus 2 (second row), negative-sequence (third column) voltage, etc..

- Write the function *Convert_Sequence2Phase_Currents ()*, which converts the sequence current array *Iseq* to the phase current array *Iph*.
- Write the function *Convert_Sequence2Phase_Voltages ()*, which converts the sequence voltage matrix *Vseq_mat* to the phase line-to-ground voltage matrix *Vph_mat*.

In the report, you are expected to briefly describe how you came up with your solution for the above functions, e.g., which equation you used and implemented in Python, etc.

Task 2 - Construction of the ***LoadNetworkData4FA()*** function

Table 3. Example of the additional required data can be handled in LoadNetworkData4FA

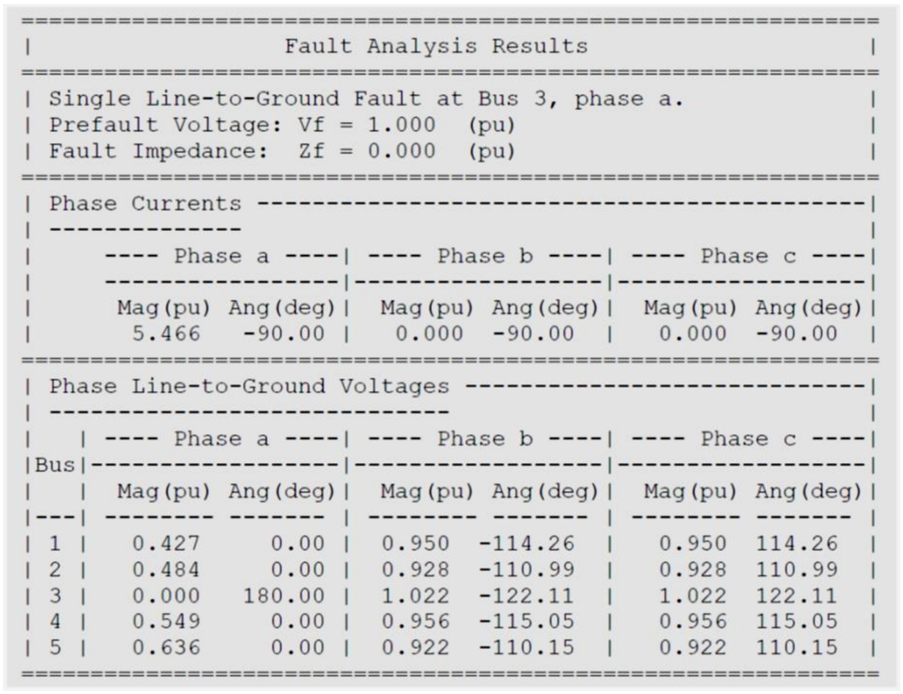
```
Inport numpy as np
import ReadNetworkData as rd
def LoadNetworkData4FA(filename):
    # include additional global variables
    global Ybus, ... , \
            Ybus0,Ybus2,Zbus0,Zbus1,Zbus2

    # read in the data from the file...
    bus_data,load_data,gen_data,line_data,tran_data,mva_base,bus_to_ind,ind_to_bus = \
    rd.read_network_data_from_file(filename)
    ##########################################################################################
    # Construct the Ybus (positive-sequence), Ybus0 (zero-sequence), and Ybus2 (negative-sequence)
    # matrices from elements in the line_data and tran_data
    # Keep/modify code from the Python power flow program as needed
    ##########################################################################################
    N = len(bus_data) # Number of busses
    Ybus = np.zeros((N,N),dtype=complex)
    Ybus0 = np.zeros((N,N),dtype=complex)
    Ybus2 = np.zeros((N,N),dtype=complex)
    # Continue with your code here...
    # ...
    # End your code by deriving the Zbus matrices (Zbus0, Zbus1, Zbus2)
    Zbus0 = np.linalg.inv(Ybus0)
    Zbus1 = np.linalg.inv(Ybus)
    Zbus2 = np.linalg.inv(Ybus2)

    return
```

Question B.2

Complete the *LoadNetworkData4FA()* function, which automatically creates the Ybus (positive-sequence), *Ybus0* (zero-sequence), and *Ybus2* (negative-sequence) matrices for the sequence networks of a given electrical network, and then derives the sequence bus impedance matrices *Zbus0*, *Zbus1*, and *Zbus2*. In your implementation, you should consider the assumptions made for the fault analysis (e.g., ignore shunts, Δ-Y phase shifts, etc.). Modifying the respective function you wrote for the Python power flow program in Assignment 1 is recommended. In the report, you are expected to describe how you came up with your answer, e.g., which equation you used, etc. Emphasis should be put on the transformer connections (your code should work for any transformer connection).

Task 3 - Construction of the **DisplayFaultAnalysisResults()** function

When the fault analysis is completed, the next step in the main script in Table 1 is to display the results. The function ***DisplayFaultAnalysisResults()***, stored in ***FaultAnalysis_46705.py***, displays phase fault currents and phase line-to-ground voltages (at all busses), for a given fault. An example of how such a printout of results could look is provided in Figure 3.



```
==============================================================
|                    Fault Analysis Results                    |
==============================================================
| Single Line-to-Ground Fault at Bus 3, phase a.               |
| Prefault Voltage: Vf = 1.000  (pu)                           |
| Fault Impedance:  Zf = 0.000  (pu)                           |
==============================================================
| Phase Currents ----------------------------------------------|
| --------------                                               |
|     ---- Phase a ----| ---- Phase b ----| ---- Phase c ----|
|     -----------------|------------------|------------------|
|      Mag(pu) Ang(deg)| Mag(pu) Ang(deg)| Mag(pu) Ang(deg)|
|       5.466   -90.00 | 0.000   -90.00  | 0.000   -90.00  |
==============================================================
| Phase Line-to-Ground Voltages -----------------------------|
| -----------------------------                              |
|   | ---- Phase a ----| ---- Phase b ----| ---- Phase c ----|
|Bus|------------------|------------------|------------------|
|   |  Mag(pu) Ang(deg)| Mag(pu) Ang(deg)| Mag(pu) Ang(deg)|
|---| -------- --------| -------- --------| -------- --------|
| 1 |   0.427     0.00 | 0.950  -114.26  |  0.950  114.26  |
| 2 |   0.484     0.00 | 0.928  -110.99  |  0.928  110.99  |
| 3 |   0.000   180.00 | 1.022  -122.11  |  1.022  122.11  |
| 4 |   0.549     0.00 | 0.956  -115.05  |  0.956  115.05  |
| 5 |   0.636     0.00 | 0.922  -110.15  |  0.922  110.15  |
==============================================================
```

*Figure 3. Example of the output of **DisplayFaultAnalysisResults()**.*

Question B.4

Use the Fault Analysis program for analyzing the following types of faults at bus 4011:

- 3-Phase Balanced fault;
- Single Line-to-Ground fault (at phase a);
- Line-to-Line fault (between phases b and c);
- Double Line-to-Ground fault (between phases b and c).

Question B.5

Verify your results with the results in PowerFactory using the provided model of the Nordic32-bus system.
We will discuss in class which modifications to the algorithm might be necessary in order to match the standard used in PowerFactory and how to set up the calculation in PowerFactory to match the assumptions in the Python algorithm.

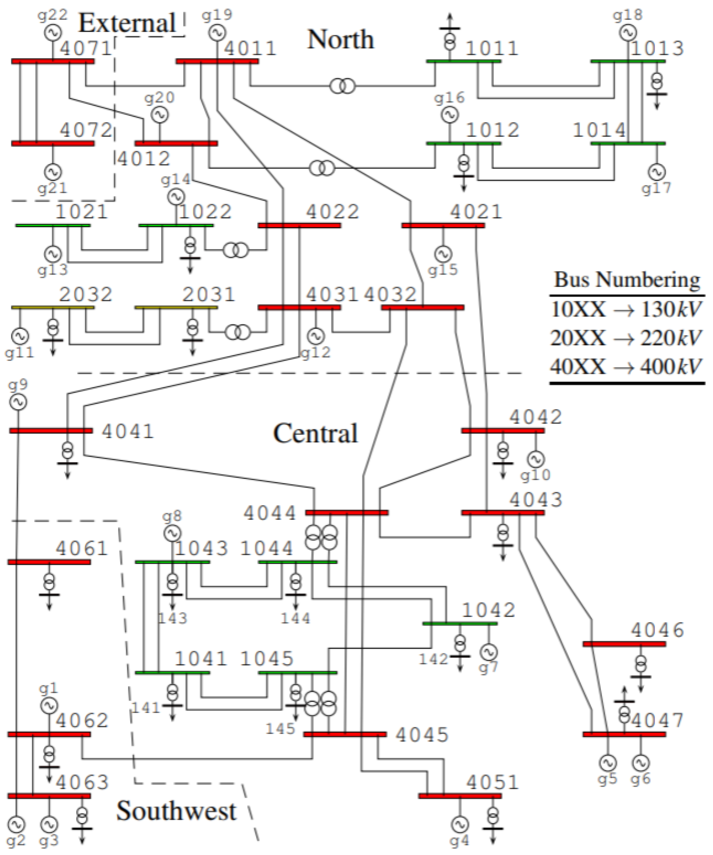


Figure 4. Nordic32-bus system topology

Part C

Question C.1

The design of the protection of more general system configurations (meshed systems) involves dividing the system into so-called protection zones. The protection zones are defined by the equipment and the available circuit breakers where six categories of protection zones are possible: (1) generators and generator-transformer units, (2) transformers, (3) buses, (4) lines (transmission, sub-transmission, and distribution), (5) utilization equipment (motors, static loads, or other), and (6) capacitor or reactor banks (when those are separately protected). Answer the following:

a) Determine the zones of protection for the system in Figure 5
b) Describe how the overlap of zones is accomplished in practice.
c) Determine which circuit breakers should open for a fault at:
   i. P_1
   ii. P_2
   iii. P_3

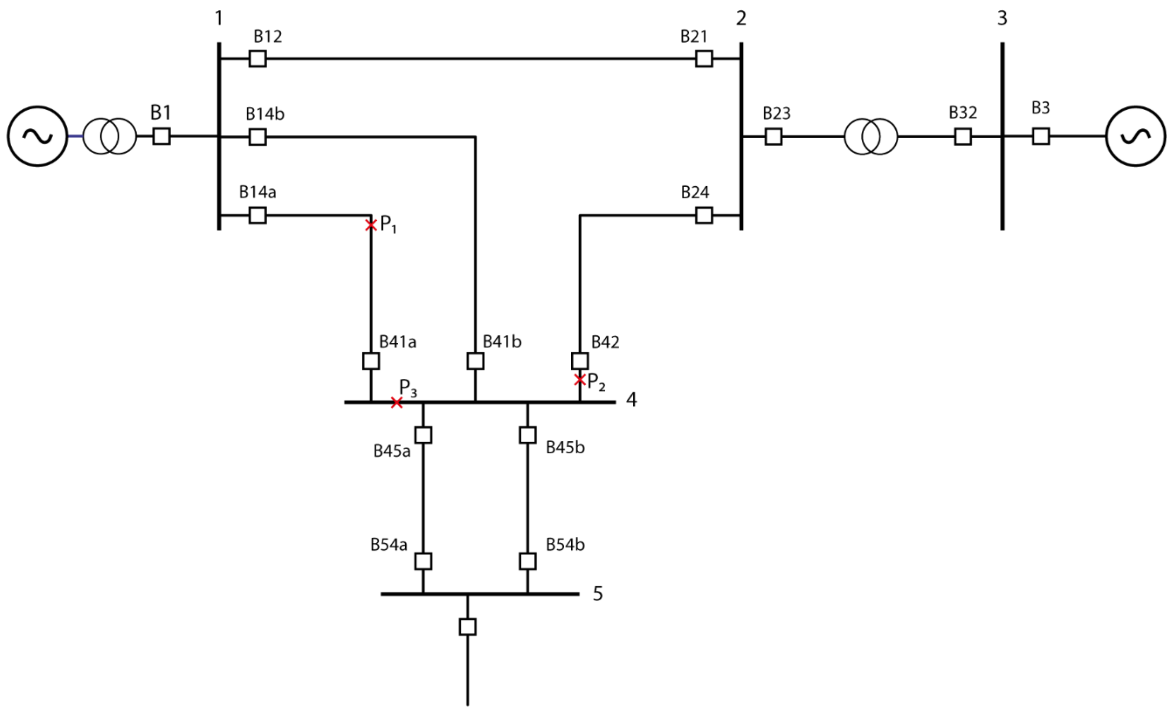


Figure 5. The system used in Q7. This figure is uploaded with the assignment text in various formats (.png, *.eps, *.pdf) that you can use when preparing your report.*

Question C.2

Answer the following related to distance relays:

a) Describe the basic principle of distance (impedance) relays

b) Determine appropriate relay settings for a three-zone directional distance relay at B12 in the meshed system in Figure 6. Three-zone mho relays protect all the lines. The positive sequence impedances of each line are provided in Table 4. The rated voltage for the high-voltage buses is 500 kV. The CT ratio is 2500:5, and the VT ratio is 4500:1 at B12. The task ahead is to determine the settings Z_{t1}, Z_{t2} and Z_{t3} for the mho relay at B12:

   1. Determine the impedance values in ohms for each zone.
   2. Determine the settings of the mho relay for the three zones Z_{t1}, Z_{t2} and Z_{t3} (the impedances seen by the relays).

c) Use Python to draw up the characteristics for the three zones, a mho relay and an impedance relay with directional restraint. (Use the same relay settings as determined in (b).) Mark in the plot the operating point corresponding to an emergency loading of 1800 A at a lagging power factor of 0.95. Check whether any of the relays will trip or not during this condition.

Table 4. Impedance values for lines connected to bus 2 in Q.8

Line	Positive sequence impedance [Ω]
1-2	7+j70
2-3	6+j60
2-4	8+j80

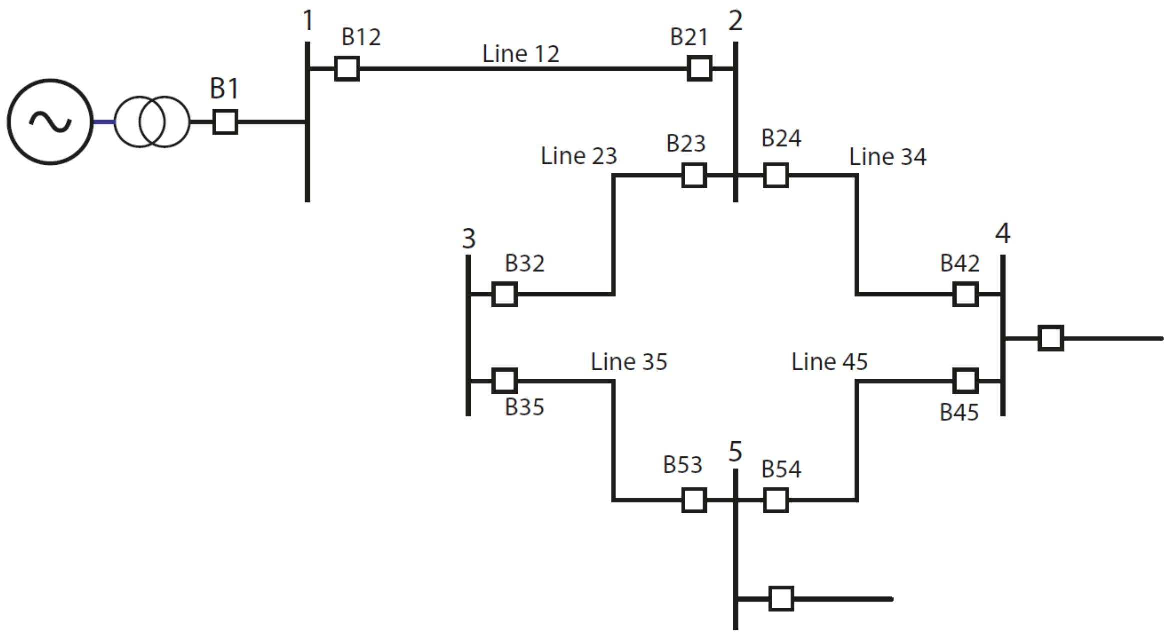


Figure 6. The system used in question C.2

Part D

Question D.1 (preliminary studies of model configuration)

Consider the following two bus system illustrated in Figure 7,

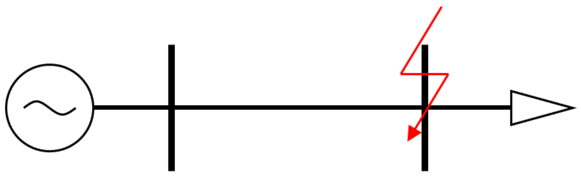


Figure 7. Single machine system

The single machine (SM) system is given in PowerFactory, named “SM_fault_analysis.pfd”.

A script supporting external control of PowerFactory is obtained, named “powerfactory_dynamic_simulation_handout.py”.

Do the following:

- Read the script and understand it by making a crude flowchart.
- Prepare the script to run EMT simulation with unbalanced network representation.
- Succeed in running PowerFactory via Python.
- Evaluate whether the script dig what you expected.
- Plot the values of a 3 phase to ground fault for 10 seconds.
- Write the dynamic fault current $i(t) = i_{ac}(t) + i_{dc}(t)$
- Compare the analytic with the PowerFactory results.

Question D.2 (Continuing with the system in Question D.1)

Point on wave (POW) sensitivity analysis is often used in power system stability analysis, as the voltage and current response depends on fault time.

Using the analytic solution, sweep through different fault inception angles to evaluate the largest peak current.

Part E

PowerFactory can be automated to perform the contingency tasks through Python script which can significantly save time compared to manually executing each contingency scenario individually. In this part, you are provided the PowerFactory model of the Nordic 32-bus system, where your task is to verify partly or fully the results you obtained from the previous assignment.

1. Verify your previous load flow assignment result with the Power Factory result for the base case. Identify the difference and clarify the results in case of minor discrepancies

2. Develop the Python script that automates the contingency analysis based on the example provided. Verify the results with the contingency analysis.