Spring
2024

46705
Power Grid Analysis

Assignment 2

Subject: Security Assessment in Python and System Protection PART I (Part II issued after next week's lecture)

Issued: ***27.02.2024***
Submission date: ***01.04.2023***

Introduction

This assignment touches upon the topics of security assessment and system protection. The assignment addresses in particular the following learning objectives defined for the course. A student that meets the objectives is able to:

- **Describe** the basic principle of power system security analysis, and **apply** steady state contingency analysis.
- **Describe** the basic principles of distance relays, **design** their zones of protection in a distance protection scheme

Specific Learning Objectives for the Assignment

A student meeting the more speific objectives of this assignment is able to:

- **Apply** security assessment by extending the previously developed Python based load-flow program so that it can carry out a contingency assessment of an arbitrary system.
- Use the program to perform security contingency analysis of the Nordic32 test system and **analyze** the results.
- **Determine** zones of protection for a given test system, and **determine** appropriate settings for distance relays.

Note: It is a good idea to address these points in your conclusion and keep them in mind when preparing for the final multiple choice exam.

Part I - Static Security Assessment

Q.1 Essay Question (max 1 page): Briefly describe the concept of power system security and describe how the security assessment can be broken down into three major functions that are carried out in system operators' control centers.

Contingency Analysis in Python

The purpose of this part of the assignment is to create a Python program that carries out a contingency analysis of a given system. To achieve this, you will need to modify your Python code from assignment 1 as well as creating new functions for the contingency analysis. The core idea behind the Python program that carries out the contingency analysis is depicted as a flow-diagram in figure 1.

Table 1 shows an example script which has implemented some of the steps in the figure. Your Python code from assignment 1 (load flow) is re-used and modified. The functions marked in red, represent the functions and library you need to write/modify.

In the previous two assignments, the systems models have been stored in *.txt files. In this assignment, the *.txt files containing the model have been extended further, such that they now contain the MVA ratings for the lines, transformers and generators. The Python file `ReadNetworkData.py`, which was uploaded with the assignment has been modified such that it now can parse the new model data (that contains the MVA ratings).

You need to modify your `LoadNetworkData.py` so it can correctly work with the new version of the `ReadNetworkData.py`. The following code segments provide hints regarding which modifications need to be made to `LoadNetworkData.py`.

```
    ################################################
    # New code to be included in LoadNetworkData()  #
    ################################################
    global Lines, Trans, Gen_MVA #new variables as global variables.
    Lines = []
    Trans = []
    Gen_MVA = np.zeros(N) #keep track of generators MVA size  (bus indices used)

    ##########################################################
    # modifications when dealing with transmission lines... #
    ##########################################################
    for line in line_data:
        bus_fr, bus_to, id_, R, X, B, X2, X0, MVA_rate = line #unpack the values
        Lines.append([bus_fr, bus_to, id_,Y_se,Y_sh_2,MVA_rate]
        ############################################
        # ..... the remaining code comes here
        #    (for the Ybus matrices ....)
        ############################################

    ##########################################################
    # modifications when dealing with transformers...       #
    ##########################################################
    for line in tran_data:
        bus_fr, bus_to, id_, R,X,n,ang1,fr_co, to_co, X2, X0, MVA_rate = line #unpack values
        Trans.append([bus_fr, bus_to, id_, Yps_mat, MVA_rate ])
        ############################################
        # ..... the remaining code comes here
        #    (for the Ybus matrices ....)
        ############################################
```

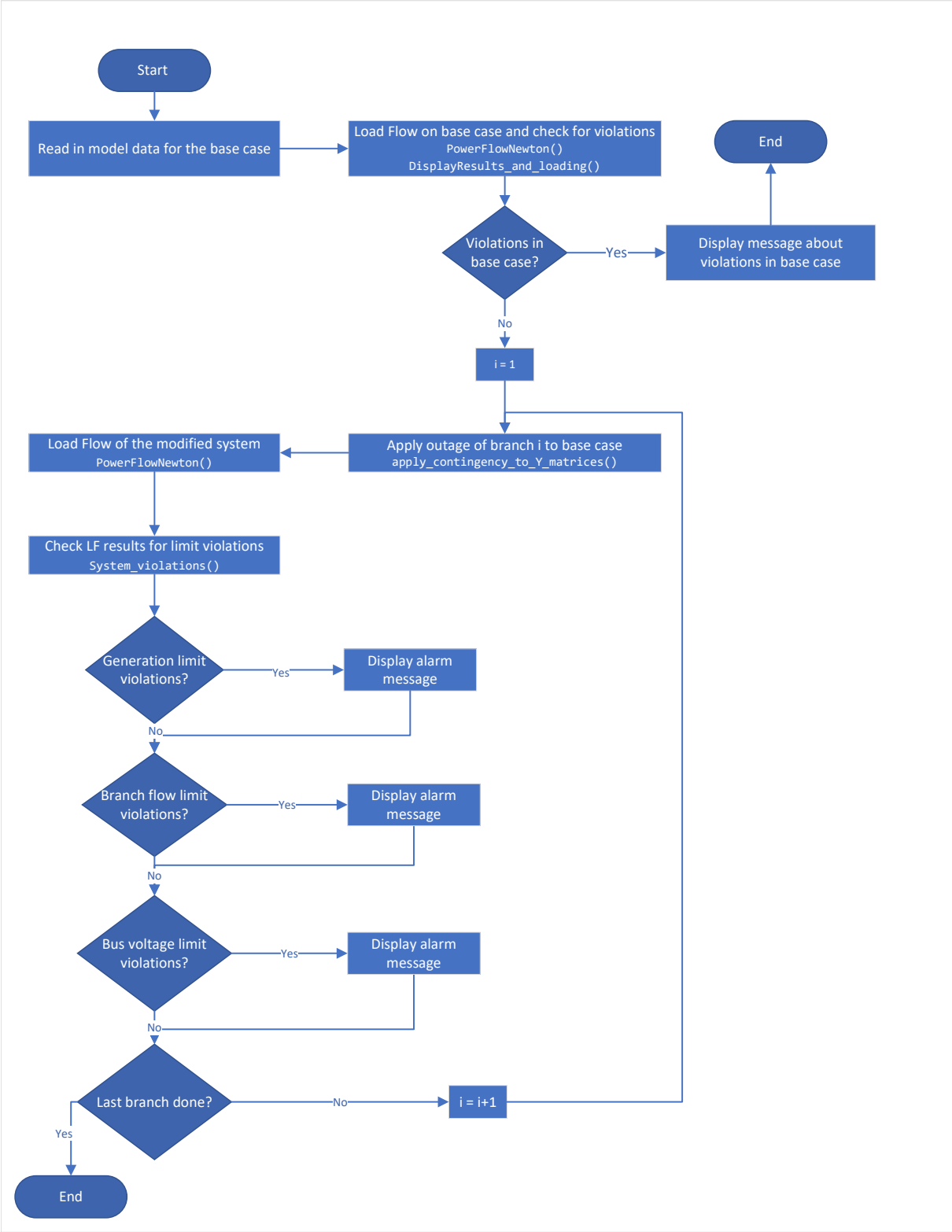

Figure 1: Flow-diagram for a proposed structure of the contingency analysis program to be created in this assignment.

```python
import numpy as np
import PowerFlow_46705 as pf          # import Power Flow functions
import LoadNetworkData_V2023 as lnd   # load the network data to global variables
max_iter = 30   # Iteration settings
err_tol = 1e-4
# Load the Network data ...
filename = "./TestSystem4SA.txt"
lnd.LoadNetworkData(filename) # makes Ybus available as lnd.Ybus etc.
#%%
######################################################################
#   Part I:  Study the base case and display results (with % loading) #
######################################################################
V,success,n = pf.PowerFlowNewton(lnd.Ybus,lnd.Sbus,lnd.V0,lnd.pv_index,
                                 lnd.pq_index,max_iter,err_tol)
if success: # Display results if the power flow analysis converged
    pf.DisplayResults_and_loading(V,lnd)

#%%
######################################################################
#  Part II:  Simplified contingency analysis (only branch outages)   #
######################################################################
print('*'*50)
print('*             Contingency Analysis               *')
print('*'*50)

for i in range(len(lnd.br_f)): #sweep over branches
    fr_ind = lnd.br_f[i]
    to_ind = lnd.br_t[i]
    br_ind = i
    Ybr_mat = lnd.br_Ymat[i]
    Ybus_mod,Yfr_mod,Yto_mod = \
                apply_contingency_to_Y_matrices(lnd.Ybus,lnd.Y_fr,lnd.Y_to,
                                                fr_ind,to_ind,br_ind,Ybr_mat)

    str_status = '-'*63 + '\nTripping of branch {:} (bus {:} - bus{:})'.format(i+1,/
                                          lnd.ind_to_bus[fr_ind],lnd.ind_to_bus[to_ind])

    try:  #try the load flow, if it fails, display message
        V,success,n = pf.PowerFlowNewton(Ybus_mod,lnd.Sbus,lnd.V0,lnd.pv_index,lnd.pq_index,/
                                         max_iter,err_tol,print_progress=False)
    except:
        str_ = ' --> Load Flow error (Jacobian) when branch {:} (bus {:} - bus {:}) is tripped'
        str_ = str_.format(i+1,lnd.ind_to_bus[fr_ind],lnd.ind_to_bus[to_ind])
        print(str_status+ ' [CONVERGENCE ISSUES!]')
        print(str_)
    else:
        if success: # Display results if the power flow analysis converged
            violations = System_violations(V,Ybus_mod,Yfr_mod,Yto_mod,lnd)
            if not violations: # no violations, print status and move on
                print(str_status + ' [OK!]')
            else: # if violation, display them
                print(str_status + ' [Violations!]')
                for str_ in violations:
                    print(str_)
        else: #no convergence...
            str_ = ' --> No load-flow convergence when branch {:} (bus {:} - bus {:}) is tripped'.
            str_ = str_.format(i+1,lnd.ind_to_bus[fr_ind],lnd.ind_to_bus[to_ind])
            print(str_status + ' [CONVERGENCE ISSUES!]')
            print(str_)
```

Table 1: Example script for carrying out contingency analysis in Python. The functions or library colored in red are the one you need to create/modify during the assignment.

Q.2 Modify your previous `DisplayResults()` function from assignment 1 in such a way that it shows the loading levels of the generators and branches in percentages (the modified function is called `DisplayResults_and_loading()`). Furthermore, extend the function such that it can appropriately handle generation and load connected to one bus. The output from the modified function could resemble the one shown in table 2.

```
================================================================================
|                                  Bus results                                 |
================================================================================
 Bus       Bus           Voltage            Generation                 Load
  nr      Label      Mag(pu)  Ang(deg)   P (pu)  Q(pu)  loading     P (pu)  Q(pu)
-----   ---------   -------  -------   -------  -------- -------   -------  -------
 1011    BUS1011     0.989     1.02      -         -       -        2.000     0.800
 1012    BUS1012     1.000     4.80    6.000     1.851    78.49%    3.000     1.000
 1013    BUS1013     1.000     9.76    3.000     0.808    51.78%    1.000     0.400
 1014    BUS1014     1.000    13.14    5.500    -0.543    78.95%      -         -
                        ........ and so on ..........

 ============================================================================
|                         Branch flow                                        |
 ============================================================================
|Branch  From    To        From Bus Injection           To Bus Injection     |
|  nr    Bus     Bus    P (pu)   Q (pu)  loading     P (pu)   Q (pu) loading|
 -----  -----  -----  -------  --------  -------     -------  ------- -------
   1    1011   1013    -2.103   0.296    42.47%      2.149   0.013   42.98%
   2    1011   1013    -2.103   0.296    42.47%      2.149   0.013   42.98%
   3    1012   1014    -1.556   0.351    31.89%      1.591  -0.139   31.95%
   4    1012   1014    -1.556   0.351    31.89%      1.591  -0.139   31.95%
                ........ and so on ..........
```

Table 2: Example showing which type of information should be printed out automatically when the modified function `DisplayResults_and_loading()` is called.

Q.3 Write the function `System_violations()` that automatically investigates whether any power flow limits or voltage limits have been violated. The function should return a list of strings that describe the found violations. Table 3 shows a template for the function.

Q.4 Write the function `apply_contingency_to_Y_matrices()` which takes as input the original admittance matrices (`Ybus`, `Y_fr` and `Y_to`) and returns the modified version of those matrices where the contingency (branch tripping) has been applied. A template for the function is provided in table 4.

```python
def System_violations(V,Ybus,Y_from,Y_to,lnd):
    # Inputs:
    # V = results from the load flow
    # Ybus = the bus admittance matrix used in the load flow
    # Y_from,Y_to = tha admittance matrices used to determine the branch flows
    # lnd = the LoadNetworkData object for easy access to other model data

    #store variables as more convenient names
    br_f=lnd.br_f; br_t=lnd.br_t;   #from and to branch indices
    ind_to_bus=lnd.ind_to_bus;      # the ind_to_bus mapping object
    bus_to_ind=lnd.bus_to_ind;      # the bus_to_ind mapping object
    br_MVA = lnd.br_MVA             # object containing the MVA ratings of the branches
    br_id  = lnd.br_id              # (you have to update LoadNetworkData for this)


    #line flows and generators injection....
    S_to = V[br_t]*(Y_to.dot(V)).conj()          # the flow in the to end..
    S_from = V[br_f]*(Y_from.dot(V)).conj()      # the flow in the from end
    S_inj = V*(Ybus.dot(V)).conj()               # the injected power
    SLD=lnd.S_LD                                 # The defined loads on the PQ busses
    S_gen = S_inj + SLD                          # the generator arrays

    violations = []   #empty list that will store strings describing each violation

 ##############################################################################
#
#    YOUR CODE COMES HERE:
#
#    1. Check flow in all branches (both ends) and report if limits are violated
#    2. Check output of all generators and see if limits are exceeded
#    3. Check voltages on all busses and see if it remains between 0.9 and 1.1 pu
#
##############################################################################

    return violations #return the list with description of all of the violations
```

Table 3: Template for the function `System_violations()` which is to be written in Q.3.

```python
def apply_contingency_to_Y_matrices(Ybus,Yfr,Yto,fr_ind,to_ind,br_ind,Ybr_mat):
    # input:
    # The original admitance matirces: Ybus,Yfr,Yto
    # The from and to end indices for the branch (fr_ind, to_ind)
    # The indice for where the branch is in the branch list (br_ind)
    # The 2x2 admittance matrix for the branch  Ybr_mat
    ########################################################

    # This is important, you must copy the original matrices
    Ybus_mod = Ybus.copy()  # otherwise you will change the original Ybus matrix
    Yfr_mod = Yfr.copy()    # when ever you make changes to Ybus_mod
    Yto_mod = Yto.copy()    # using the .copy() function avoids this

##############################################################################
#
#    YOUR CODE COMES HERE:
#
#    1. Remove the branch from the Ybus_mod matrix
#    2. Remove the branch from the Yto and Yfr matrices
#
##############################################################################

    return Ybus_mod,Yfr_mod,Yto_mod
```

Table 4: Template for the function `apply_contingency_to_Y_matrices(()` which is to be written in Q.4.

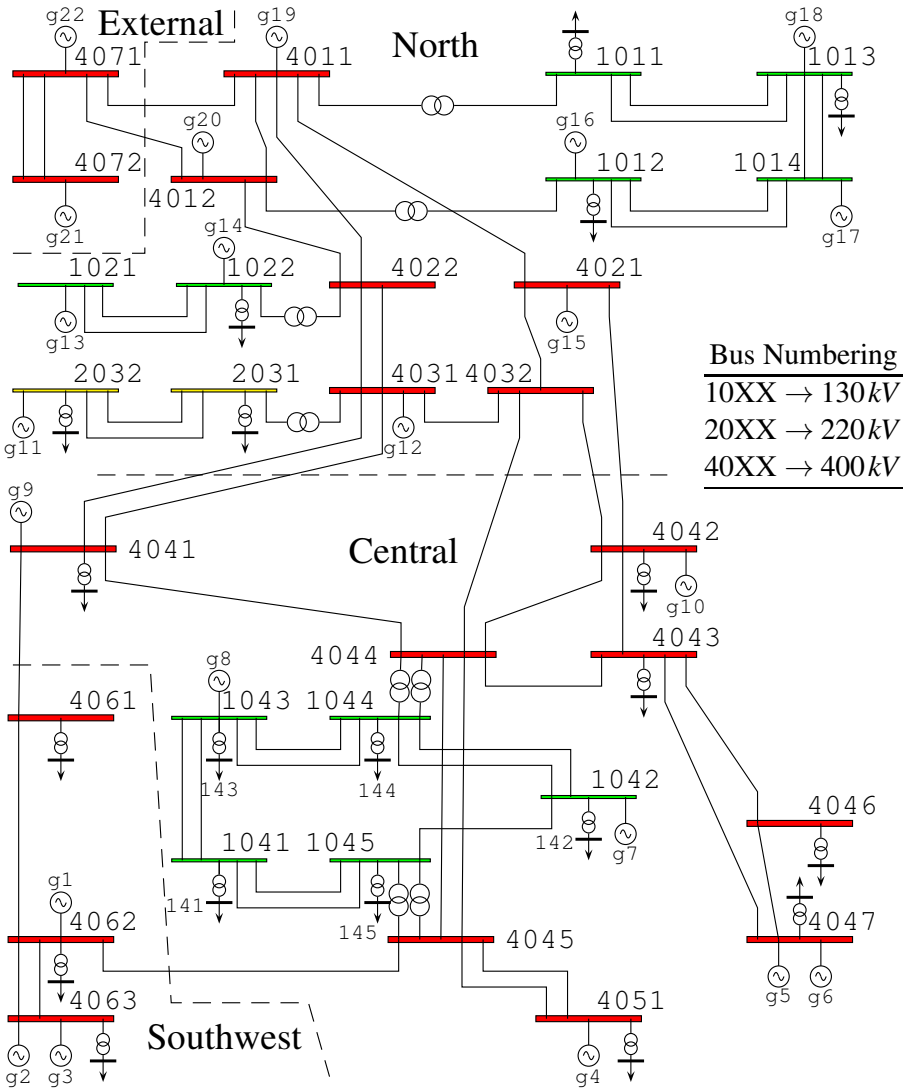

Figure 2: One-line diagram of the Nordic32 system studied in Q.5 and Q.6.

Q.5 Load the Nordic32 model (Nordic32_SA.txt) and run the load flow. Examine whether there are any flow limit or voltage limit violations in the base case. Figure 2 shows the topology of the Nordic32 system.

Q.6 Run the contingency analysis study the results. Give examples of contingency that result in no violations, contingencies that result in voltage limit violations, and contingencies that result in no load-flow convergence.

Part II - Coordinated System Protection

Part II will be revealed after next week's lecture.

Guidelines for the Writing of the Report

When writing the report for the assignment, the below listed points should be followed:

- The report must start with a self-evaluation section, where the authors state whether the report represents an even contribution from all group members or whether some participants were not effectively contributing to the report.
- The front page of the report should contain your names and student numbers.
- Your Python/Matlab code used for the solution of this assignment has to be included in appendix.
- The report shall contain the answer to each of the questions. The answers shall be clearly separated from each other and come in the same order as the questions in the assignment. It is not necessary to repeat the text from the assignment in your report.
- When answering the questions, both the results as well as an explanation on how you came up with your results should be included in the answer.
- Always when asked to provide plots of something (time responses, curves etc.) you must provide your interpretation of the figure.
- Make sure that the assignment's learning objectives are reflected in your report.

- When your reports are evaluated, points are awarded for the general setup of the report. We are looking for a professional look of the report, where among others the following is considered:
  - Readability (appropriate font sizes, margins etc.) and consistence in styles applied (same look for body text, headings, captions etc. throughout the report).
  - Presentation of equations should be of an appropriate quality (do not copy/paste equations as screenshot pictures from the lecture handouts).
  - Plots and figures should be of good quality (no fuzzy looking screenshots in the report). You can always ask the teachers how you can export figures of good quality in Python/Matlab.
  - Figures and tables must have a label and caption (e.g. *Figure 1: plot of active power in respect to* ....).
  - If you are citing external material, you need to include a properly set-up reference list (you can use any citation style you prefer).
  - Does the report contain a conclusion section?
- You will also be awarded points for the general formulation of the report where focus is on:
  - The flow in the text when explaining how you came up with your answers and correct use of relevant terminology.
  - Interpretation and reflection of results when appropriate.
  - Grammar and spelling.

Guidelines for the Upload of the Report and Python Code

The report submission is carried out on DTU-LEARN where the following shall be uploaded:

- PDF version of your report, with your Python code in appendix.
- Your Python code used for solving the assignment (the *.py). The code shall include your comments where you explain the meaning behind the code.