Collin Griffin
Dr. Kerestes
ECE 2774
29 March 2024

Milestone 3 A

Repository

Link: https://github.com/griffincj/Power-System-Analysis

Overview

This milestone adds the foundation for solving the power flow problem using the Newton-Raphson algorithm. Specifically, this portion of the milestone adds methods required for calculating the Jacobian matrix, and for performing the steps of the algorithm needed to generate the properly sized power mismatch and delta-x vectors.

Main Module

The output required for this module can be displayed by running the main method of the main.py module. This should output the four quadrants of the Jacobian matrix, the delta_y matrix, and the delta_x matrix. Both delta matrices are 11x1 in their calculated form. The delta_x matrix has a supplementary form available called "x_full", which is sized to be 14 x 1, and includes entries for the slack and PV buses.

The module was updated with calls to create the Loads and Generators needed to initialize the power system and the necessary values on the buses power, reactive power, voltage, and angle attributes.

Generator and Load Classes

- The generator and load classes have been added in this milestone
- They are almost identical, except for a few key differences:
    - The generator sets its connected bus's type to "PV" unconditionally, whereas the load class sets the bus type to "PQ" only if it is not already a "PV" bus
    - The load class subtracts real and reactive power from the bus. The generator adds power and sets the voltage of the connected bus

PowerSystem

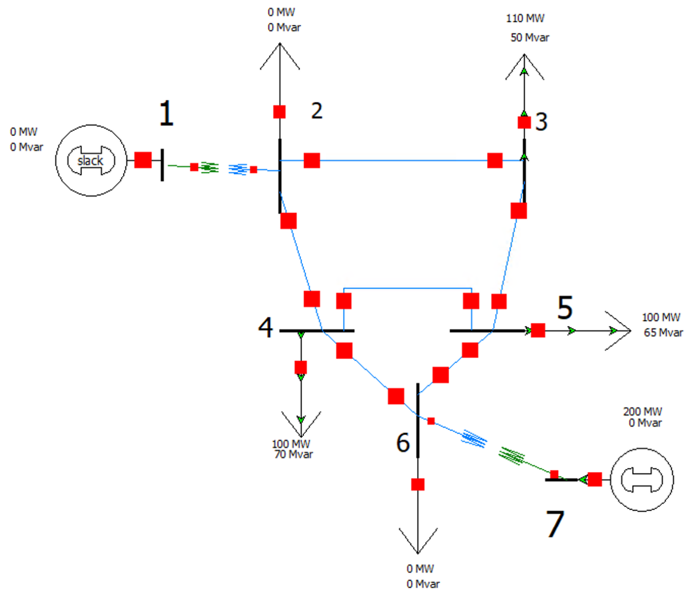
The power system class has been extended to add several new methods for solving the power flow problem.

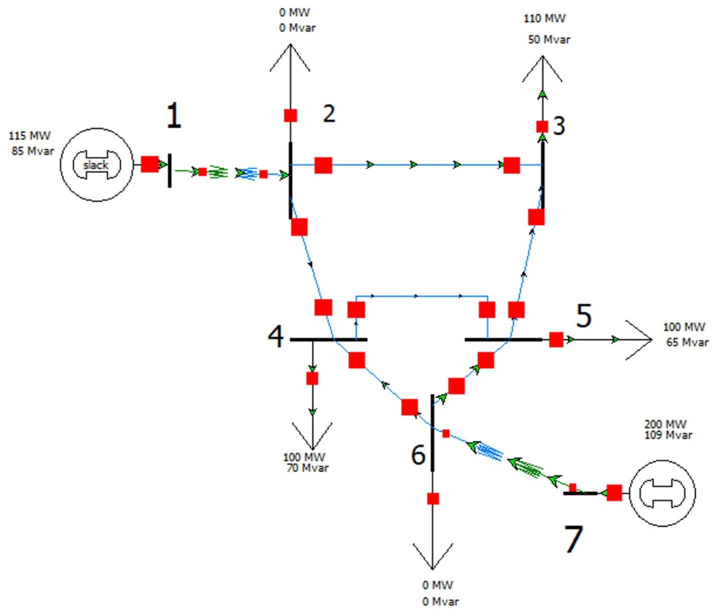
Methods

- Init_jacobian(self)
    - Sets the j1, j2, j3, and j4 Jacobian quadrants to their proper sizes by determining the number and location of the slack and PV buses
- Calc_jacobian_quad_{1-4}
    - Four separate methods, each largely similar but with separate equations for their respective Jacobian quadrant
    - Iterates over the buses for the quadrant as declared by the init_jacobian() method
    - Stores values of each calculation in Pandas DataFrame, so that values can be indexed using their bus.bus_name attribute
    - Note that each quadrant is calculated using a separate method and is stored in a separate DataFrame. In the execution of the Newton-Raphson query, the quadrants are concatenated together and converted to a Numpy matrix in order to use linear algebra to solve the power flow problem.
- Run_newton_raphson(self, iterations)
    - Method that currently implements the calculation of the delta_y and delta_x mismatch matrices
    - First, delta_y (power mismatch) is calculated after the Jacobian is assembled from the separately calculated quadrants
    - Second, the delta_x (voltage angle and magnitude vector) is calculated from taking the product of the inverse Jacobian and the delta_y vector.
    - Both vectors are 11 x 1 in our problem. A “x_full” vector is also generated, which contains placeholder values for slack and PV bus attributes that are not calculated in the algorithm.
        - The program sets these to flot(‘-inf’) for the duration of the algorithm. Ultimately, the powers for the buses will calculated after the model has theoretically converged.

PowerWorld Comparison

Pictured is the model in PowerWorld before and after running the Netwon-Raphson solver. I included the initial PowerWorld output for the Jacobian in the milestone 3 repository, in the “writeups” Directory. My program’s output appears to match aside from a from a few significant figures/rounding (the export from PowerWorld was only to the second decimal place).


0 MW
0 Mvar
110 MW
50 Mvar
1
2
3
0 MW
0 Mvar
slack
4
5
100 MW
65 Mvar
200 MW
0 Mvar
100 MW
70 Mvar
6
7
0 MW
0 Mvar


0 MW
0 Mvar
110 MW
50 Mvar
1
2
3
115 MW
85 Mvar
slack
4
5
100 MW
65 Mvar
200 MW
109 Mvar
100 MW
70 Mvar
6
7
0 MW
0 Mvar