Milestone 3b Documentation

Github repository: <https://github.com/Kamdens111/ECE2774_Project>

**I. Class Documentation**

**Settings Class:**

NOTE: USED TO SET THE SYSTEM MVA AND FREQUENCY

**Bus Class:**

Syntax:

bus\_name = Bus(bus\_name: str, nominal\_bus\_voltage: float)

Input Variables:

bus\_name: string that assigns the name of the bus

nominal\_bus\_voltage: float or double that provides nominal voltage that the bus operates at

Accessible class functions:

* set\_bus\_voltage(bus\_v)
  + This function assigns voltage to the bus. This will be used to assign the bus voltage after the power flow has been conducted. This will NOT change the assigned nominal voltage
* show\_bus\_numer()
  + This function will show the assigned bus number based on the order in which the busses were instantiated. The bus number starts at an index of 0 and increments by values of 1 each time a new bus is created.
* show\_bus\_count()
  + This function prints the total number of busses that have been created.

**Transformer Class:**

Syntax:

tx\_name = Transformer(tx\_power\_rating: float, percent\_z: float, x\_r\_ratio: float, busA: Bus, busB: Bus)

Input Variables:

tx\_power\_rating: float value that gives the power rating of the transformer

percent\_z: float value given as a percent (not a decimal) that gives the percent impedance of transformer

x\_r\_ratio: float value that gives the reactance over resistance ratio

busA: this is the first bus connection to the transformer

busB: this is the second bus connection to the transformer

Accessible class functions:

* show\_params()
  + This prints the resistance and reactance calculated in per unit
* get\_bus\_admittance()
  + Returns the primitive Y bus for the transformer

Non-accessible class functions:

* \_\_calc\_params()
  + This function is called in constructor and is performed as the creation of the transformer object. This function calculated the old per unit transformer parameters by using the equation:
  + The new per unit impedance is the calculated by using the equation:
  + The Y bus is also calculated by finding R and X by finding the real and imaginary parts of Z = R + j\*X
  + Since Y = 1/Z, the primitive Y bus is calculated to be
  + NOTE: BUS VOLTAGES ARE ASSUMED NOMINAL VOLTAGES FOR TRANSFORMER

**Geometry Class:**

Syntax:

geometry\_name = Geometry(d\_ab: float, d\_bc: float, d\_ca: float, conductor\_distance: float, num\_conductors: float)

Input variables:

d\_ab: float that gives distance in feet between phase A conductor(s) and phase B conductor(s)

d\_bc: float that gives distance in feet between phase B conductor(s) and phase C conductor(s)

d\_ca: float that gives distance in feet between phase C conductor(s) and phase A conductor(s)

conductor\_distance: distance in feet between the conductors in each phase for each phase, must have equilateral distances

num\_conductors: number of conductors in each bundling

Non-accessible class functions:

* \_\_calc\_params()
  + This function internally calculates the Deq that is needed in the TransmissionLine class using the equation:

**Conductor Class:**

Syntax:

Conductor\_name = Conductor(GMR: float, R\_c: float, d\_conductor: float, geometry: Geometry)

Input variables:

GMR: this is a float value that is taken from datasheet for GMR in feet

R\_c: this is float value that is resistance of the conductor in ohms/mile

d\_conductor: this is diameter of the conductor given in inches from datasheet

geometry: this is an object from the geometry class that has already been instantiated.

Non-accessible class functions:

* \_\_calc\_params()
  + This function takes values from the geometry class as well as values passed in the conductor class to calculate the Dsl and the D­sc for the TransmissionLine class.
  + This function is referenced in the constructor and performs the calculations as soon as an instance of the Conductor class is created.
  + Uses if statements that will change equations based on number of conductors in bundle.
  + Equations used:

|  |  |  |
| --- | --- | --- |
| N | Dsl | Dsc |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

**TransmissionLine Class:**

Syntax:

T\_line\_name = TransmissionLine(line\_length: float, conductor: Conductor, busA: Bus, busB: Bus)

Input variables:

Line\_length: length of transmission line in miles

Conductor: this is conductor object that already has an instance created

busA: this is the first bus connection to the transmission line

busB: this is the second bus connection to the transmission line

NOTE: BUS NOMINAL VOLTAGES MUST BE SAME BETWEEN TRANSMISSION LINE

Accessible class functions:

* get\_Z\_pu()
  + returns the per unit impedance value
* get\_Y\_pu()
  + returns the per unit shunt susceptance
* get\_series\_admittance()
  + returns the series admittance in per unit
* get\_bus\_admittance()
  + returns the primitive Y bus
  + calculated to be:
  + this function is called in the constructor

Non-accessible class functions:

* \_\_calc\_params()
  + This function calculates the per mile impedance and susceptance values using the following equations:
    - G’=0
    - R’=Rc­­/num\_of\_condcutors
    - X’ =
    - B’ =
  + The these values are then multiplied by the line length to get the actual values which are used to get the bus admittance
  + This function is called in constructor and is performed upon creation of instance.

**PowerFlow Class:**

Syntax:

Name = PowerFlow(name: str)

Input variables:

name: This is a string variable assigning name to the simulation

Accessible class functions:

* Add\_bus(bus\_name: str, bus\_voltage)
  + Creates busses and the nominal voltages associated with each bus
  + MUST CREATE BUS BEFORE ADDING COMPONENTS TO BUS
* Add\_transformer(name, power\_rating, percent\_z, x\_r\_ratio, busA: str, busB: str)
  + Creates transformer with given parameters
  + Must use strings that match busses previously made
* Add\_geometry(name: str, d\_ab, d\_bc, d\_ca, conductor\_distance, num\_conductors)
  + Creates geometry of bundles which will be passed to conductor
* Add\_conductor(name, GMR, R\_c, d\_conductor, geometry\_name: str)
  + Passes previously made geometry which will be passed to transmission line
* Add\_transmissionLine(name, line\_length, conductor\_name: str, busA: str, busB:str)
  + Creates transmission line between busses
  + Must have already created busA and busB and pass as strings
* Get\_y\_bus()
  + Returns the complete y bus of the system
* Calc\_jacobian(J1, J2, J3, J4)
  + Calculates jacobian and sizes correctly

Calc\_mismatch(V)

* + Correctly sizes the mismatch with correct size
* Calc\_solution(y, J)
  + Calculates the solution vector with correct size
* Calc\_line\_currents(V)
  + Calculates the currents for all lines
  + Input is solution voltage vector
* Calc\_power\_flow(V)
  + Calculates the power injected at each bus
  + Calculates in per unit
* Calc\_line\_losses()
  + Calculates the losses for each line

Non-accessible class functions:

* N/A

**Output Confirmation:**

PowerWorld simulation:

A diagram of a diagram

Description automatically generated

Voltages: Power World

A screenshot of a computer

Description automatically generated

Code Bus Voltages (per unit):

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated