task1

August 23, 2022

1 Task 1: Nodal ecuations

1.1 Name: Sergio Andres Vargas Mendez

Calculate the admittance and impedance matrix from the circuit. Then, get the currents, powers and losses (by differents methods).

$Z_l = 0.0013 + j0.003[pu/km]$

$B_c = j0.0008[pu/km]$

1.2 Modules

Methods will be explained when solution's instance call it, first modules have to be imported.

[1]:
```
#Append directory source code
import sys
sys.path.append("/home/FoxyLuxe/Repo/tasks/power_circuits/src/structures")

#Objects models
from circuits import PowerCircuit
from nodes import Node
from lines import PILine
```

In order to solve the problem, class based circuit was made to represent power circuit as a data structure and all methods required within main circuit class to accompany the computation of admittances, impedances, currents and powers. The ordered classes are built like the next image.

1.3 Initialization

The next step is to define the node map data structure like a dictionary of index keys containing the self index, the voltage of each node and the model for connection between another ones. By the way, the connection structure is a dictionary which map every key with a node of the circuit and the value can be whatever line model (in this case is PI line model described by a distance).

[2]:
```
zl = 0.0013+0.003j
bc = 0+0.0008j

node_map = {
    1: Node(
```

```
            _index = 1,
            _voltage = (1.045, 0),
            _connection = {
                 4: PILine(_distance=190, _Zl=zl, _Bc=bc),
                 2: PILine(_distance=130, _Zl=zl, _Bc=bc),
                 3: PILine(_distance=135, _Zl=zl, _Bc=bc),
            }
        ),
    2: Node(
            _index = 2,
            _voltage = (0.982, -2.8),
            _connection = {
                1: PILine(_distance=130, _Zl=zl, _Bc=bc),
                4: PILine(_distance=100, _Zl=zl, _Bc=bc),
            }
        ),
    3: Node(
            _index = 3,
            _voltage = (0.97, -5.5),
            _connection = {
                1: PILine(_distance=135, _Zl=zl, _Bc=bc),
                5: PILine(_distance=150, _Zl=zl, _Bc=bc),
            }
        ),
    4: Node(
            _index = 4,
            _voltage = (0.96, -4.2),
            _connection = {
                1: PILine(_distance=190, _Zl=zl, _Bc=bc),
                2: PILine(_distance=100, _Zl=zl, _Bc=bc),
                5: PILine(_distance=110, _Zl=zl, _Bc=bc),
            }
        ),
    5: Node(
            _index = 5,
            _voltage = (1.08, -1.2),
            _connection = {
                4: PILine(_distance=110, _Zl=zl, _Bc=bc),
                2: PILine(_distance=120, _Zl=zl, _Bc=bc),
                3: PILine(_distance=150, _Zl=zl, _Bc=bc),
            }
        ),
}
```

Then, the instance of the circuit is created from the node mapping.

[3]:
```
circuit = PowerCircuit(node_map=node_map)
```

1.4 Admittance and Impedance matrix

To compute the admittance matrix Y_{bus} is required two conditions in index i (node i), j (connection node j). When $i = j$ the admitance should be the sum of all connections with the actual node.

$$Y_{ii} = \sum_{k=1}^{N} y_{ik}$$

But, if condition $i! = j$ is raised, the admitance is the negative value between both of them.

$$Y_{ij} = -y_{ij}$$

[4]:
```
circuit.compute_admittance()
print(circuit.admittanceMT)
```

```
[[ 2.47630822-5.71455744j -0.9354537 +2.1587393j  -0.90080726+2.07878599j
  -0.64004727+1.47703215j  0.        +0.j        ]
 [-0.9354537 +2.1587393j   2.1515435 -4.96510038j  0.        +0.j
  -1.2160898 +2.80636109j  0.        +0.j        ]
 [-0.90080726+2.07878599j  0.        +0.j          1.7115338 -3.94969338j
   0.        +0.j         -0.81072654+1.87090739j]
 [-0.64004727+1.47703215j -1.2160898 +2.80636109j  0.        +0.j
   2.96167325-6.83463059j -1.10553619+2.55123735j]
 [ 0.        +0.j         -1.01340817+2.33863424j -0.81072654+1.87090739j
  -1.10553619+2.55123735j  2.92967089-6.76077898j]]
```

The impedance matrix Z_{bus} is the inverse of Y_{bus}

$$Z_{bus} = Y_{bus}^{-1}$$

[5]:
```
circuit.compute_impedance()
print(circuit.impedanceMT)
```

```
[[-5.75963815e+14+7.67951754e+14j -7.11901496e+14+9.49201994e+14j
  -4.60044085e+14+6.13392113e+14j -5.40431955e+14+7.20575940e+14j
  -3.31244384e+14+4.41659179e+14j]
 [-5.75963815e+14+7.67951754e+14j -7.11901496e+14+9.49201994e+14j
  -4.60044085e+14+6.13392113e+14j -5.40431955e+14+7.20575940e+14j
  -3.31244384e+14+4.41659179e+14j]
 [-5.75963815e+14+7.67951754e+14j -7.11901496e+14+9.49201994e+14j
  -4.60044085e+14+6.13392113e+14j -5.40431955e+14+7.20575940e+14j
  -3.31244384e+14+4.41659179e+14j]
 [-5.75963815e+14+7.67951754e+14j -7.11901496e+14+9.49201994e+14j
  -4.60044085e+14+6.13392113e+14j -5.40431955e+14+7.20575940e+14j
  -3.31244384e+14+4.41659179e+14j]
 [-5.75963815e+14+7.67951754e+14j -7.11901496e+14+9.49201994e+14j
```

```
 -4.60044085e+14+6.13392113e+14j -5.40431955e+14+7.20575940e+14j
 -3.31244384e+14+4.41659179e+14j]]
```

1.5 Injected current

The injected current can be expressed like:

$$I_{ii} = \sum_{k=1}^{N} Y_{ik} E_k$$

[6]:
```
circuit.compute_injectedI()
print(circuit.injected_current)
```

```
[[ 0.58833711-0.25945583j]
 [-0.07243352+0.05513759j]
 [-0.48908027+0.23811863j]
 [-0.50797816+0.38227205j]
 [ 0.64070749-0.62175382j]]
```