Joe Stanley

ECE523 - HWK5

In [1]:

```python
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import electricpy as ep
from electricpy.constants import *
from tabulate import tabulate as table
```

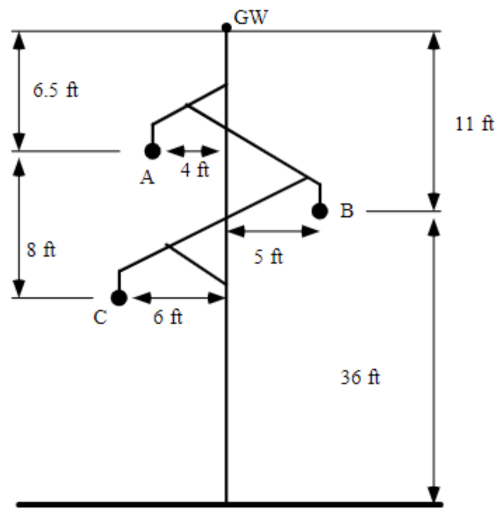
1. Compute the per mile positive and negative sequence impedance for the line configuration of figure below where the conductor is 336,400 CM, 26/7 Strand ACSR.
Ignore the ground wire for problems 1-4.

Conductor data from table:

$GMR := 0.0244ft$ $\quad$ $diameter := 0.721in$

$Rac := 0.278\frac{ohm}{mi}$ $\quad$ at 25C and 60Hz

Assume each conductor is 10 feet lower at mid span than at tower.



In [2]:

```python
# Define Parameters
GMR = 0.0244  # feet
dia = 0.721   # inches
Rac = 0.278   # ohm/mi
tmp = 25      # °C
frq = 60      # Hz

# Calculate Rd and Rself
Rd = carson_r * frq
Rself = Rac + Rd
print("Rd:",Rd,"Ω\tRself:",Rself,"Ω")

# Calculate Dab, Dbc, and Dca
Dab = np.sqrt((11-6.5)**2+(4+5)**2)
Dbc = np.sqrt((5+6)**2+(8+6.5-11)**2)
Dca = np.sqrt((6-4)**2+(8)**2)
print("Dab:",Dab,"\tDbc:",Dbc,"\tDca:",Dca)

# Calculate Zperlength Matrix
De = ep.de_calc('Avg')
Zperlen = ep.zperlength(Rd=Rd,Rself=Rself,De=De,Dab=Dab,Dbc=Dbc,Dca=Dca,Ds=GMR)
print("\nZ-per-unit Matrix:")
print(table(np.asarray(np.around(Zperlen,6),dtype=str),tablefmt="fancy_grid"))

# Calculate Sequence Impedances
print("\nSequence Impedance Matrix:")
print(table(np.asarray(np.around(ep.sequencez(Zperlen,resolve=True,round=5),6),dtype=str),tablefmt="fancy_grid"))
print("\nSequence Impedances:")
Z0, Z1, Z2 = ep.sequencez(Zperlen,resolve=True,diag=True,round=5)
print("Z1:",Z1,"Ω","\tZ2:",Z2,"Ω")
```

```
Rd: 5.9214e-05 Ω        Rself: 0.278059214 Ω
Dab: 10.062305898749054         Dbc: 11.543396380615196         Dca: 8.246211251235321

Z-per-unit Matrix:
```

(0.278059+0.000878j)	(5.9e-05+0.000424j)	(5.9e-05+0.000439j)
(5.9e-05+0.000424j)	(0.278059+0.000878j)	(5.9e-05+0.000414j)
(5.9e-05+0.000439j)	(5.9e-05+0.000414j)	(0.278059+0.000878j)

```
Sequence Impedance Matrix:
```

(0.27818+0.00173j)	1e-05j	(-0+1e-05j)
(-0+1e-05j)	(0.278+0.00045j)	(-1e-05-1e-05j)
1e-05j	(1e-05-1e-05j)	(0.278+0.00045j)

```
Sequence Impedances:
Z1: (0.278+0.00045j) Ω  Z2: (0.278+0.00045j) Ω
```

2.

Compute the phase impedance matrix Zabc for the line described in problem 1. Assume that the line is 70 miles long and is not transposed. Ignore the ground wire. Calculate the sequence impedance matrix.

In [3]:

```
# Define Line Length
LineLen = 70 # mi

# Use Existing Zperlen, find Zline
Zline2 = LineLen * Zperlen
print("Zline:")
print(table(np.asarray(np.around(Zline2,6),dtype=str),tablefmt="fancy_grid"))

# Find Zsequence
Zseq = ep.sequencez(Zline2)
print("\nZsequence:")
print(table(np.asarray(np.around(Zseq,6),dtype=str),tablefmt="fancy_grid"))
```

Zline:

(19.464145+0.061469j)	(0.004145+0.029685j)	(0.004145+0.030736j)
(0.004145+0.029685j)	(19.464145+0.061469j)	(0.004145+0.028961j)
(0.004145+0.030736j)	(0.004145+0.028961j)	(19.464145+0.061469j)

Zsequence:

(19.472+0.121j)	(-0+0j)	0j
0j	(19.46+0.032j)	(0.001-0.001j)
(-0+0j)	(-0.001-0.001j)	(19.46+0.032j)

3. Compute the total impedance matrix Zabc for the lines of problem 2 with the following transposition arrangements. Calculate the sequence impedance matrix for each.

	Fraction	*Configuration*
(a)	f1 = 0.20	a-b-c
	f2 = 0.80	b-c-a
	f3 = 0.00	c-a-b
(b)	f1 = 0.30	a-b-c
	f2 = 0.60	c-a-b
	f3 = 0.10	c-b-a
(c)	f1 = 1/3	a-b-c
	f2 = 1/3	c-a-b
	f3 = 1/3	b-c-a

In [6]:

```python
# Define Transposition Matrix Formula
def transposed_line(Z123,f1=1/3,f2=1/3,f3=1/3):
    Rp = np.array([
        [0,0,1],
        [1,0,0],
        [0,1,0]
    ])
    _Rp = np.linalg.inv(Rp)
    Zeq = f1*Z123 + f2*(_Rp.dot(Z123.dot(Rp))) + f3*(Rp.dot(Z123.dot(_Rp)))
    return(Zeq)

# a)
f1 = 0.20 # a-b-c
f2 = 0.80 # b-c-a
f3 = 0.00 # c-a-b
Zeq = transposed_line(Zline2,f1,f2,f3)
# Calculate Sequence Impedances
print("\na) Equivalent Impedance:")
print(table(np.asarray(np.around(ep.sequencez(Zeq,resolve=True,round=5),6),dtype=str),tablefmt="fancy_grid"))

# b)
f1 = 0.30 # a-b-c
f2 = 0.60 # c-a-b -> Same as a-b-c
f1 += f2
f2 = 0
f3 = 0.10 # c-b-a
Zeq = transposed_line(Zline2,f1,f2,f3)
# Calculate Sequence Impedances
print("\nb) Equivalent Impedance:")
print(table(np.asarray(np.around(ep.sequencez(Zeq,resolve=True,round=5),6),dtype=str),tablefmt="fancy_grid"))


# c)
f1 = 1/3 # a-b-c
f2 = 1/3 # c-a-b
f3 = 1/3 # b-c-a
Zeq = transposed_line(Zline2,f1,f2,f3)
# Calculate Sequence Impedances
print("\nc) Equivalent Impedance:")
print(table(np.asarray(np.around(ep.sequencez(Zeq,resolve=True,round=5),6),dtype=str),tablefmt="fancy_grid"))
```

a) Equivalent Impedance:

(19.47243+0.12106j)	(0.00023-0.00029j)	(-0.00023-0.00029j)
(-0.00023-0.00029j)	(19.46+0.03167j)	(-0.00046+0.00059j)
(0.00023-0.00029j)	(0.00046+0.00059j)	(19.46+0.03167j)

b) Equivalent Impedance:

(19.47243+0.12106j)	(0.00022+0.00038j)	(-0.00022+0.00038j)
(-0.00022+0.00038j)	(19.46+0.03167j)	(-0.00044-0.00076j)
(0.00022+0.00038j)	(0.00044-0.00076j)	(19.46+0.03167j)

c) Equivalent Impedance:

(19.47243+0.12106j)	0j	0j
0j	(19.46+0.03167j)	(-0-0j)
-0j	(-0+0j)	(19.46+0.03167j)

4 Consider the line configuration shown in the figure for problem 1. Instead of using a single conductor of 336,400 CM ACSR in each phase, with current carrying capacity of 530 amperes, suppose that each phase consists of a two-conductor bundle of two 3/0 ACSR conductors with capacity of 300 amperes/conductor. Let the two conductors of each bundle be seperated by 1.0ft vertically. Assume same sag as for problem 1.

(a) Compute the 6x6 phase impedance matrix Zabc for the bundled conductor configuration and reduce it to the 3x3 equivalent and compare with the previous solution (problem 2).

$$\mathrm{Rac4} := 0.560\,\frac{\mathrm{ohm}}{\mathrm{mi}} \qquad \text{from table}$$

$$\mathrm{GMR4} := 0.01404\mathrm{ft} \qquad \mathrm{diameter4} := 0.502\mathrm{in}$$

(b) Calculate geometric mean radius of the bundle and use the 3x3 matrix method. The is an approximation of the 6x6 matrix approach. Compare the results to part (a).

(c) Compute the sequence impedance matrix for part (a) and compare to problem 2.

In [11]:

```
# Define Given Parameters
Rac4 = 0.560
GMR4 = 0.0140
dia4 = 0.502

# Calculate Rd, Rself, and R'
Rd = carson_r * frq
Rself = Rac4 + Rd
print("Rd:",Rd,"Ω\tRself:",Rself,"Ω")
Rp = np.full((6,6),Rd) + np.identity(6)*Rac4

# Calculate Distances
De = ep.de_calc('Avg')
Ds = GMR4
Da1a2 = Db1b2 = Dc1c2 = 1.0
Da1b1 = np.sqrt((11-6.5)**2+(4+5)**2)
Da1b2 = Da1b1 + Db1b2
Da2b1 = Da1b1 - Da1a2
Da2b2 = Da1b1
Db1c1 = np.sqrt((5+6)**2+(8+6.5-11)**2)
Db1c2 = Db1c1 + Dc1c2
Db2c1 = Db1c1 - Db1b2
Db2c2 = Db1c1
Dc1a1 = np.sqrt((6-4)**2+(8)**2)
Dc1a2 = Dc1a1 - Da1a2
Dc2a1 = Dc1a1 + Dc1c2
Dc2a2 = Dc1a1

# Calculate L' Matrix
L = np.array([
    [Ds,Da1b1,Dc1a1,Da1a2,Da1b2,Dc2a1],
    [Da1b1,Ds,Db1c1,Da2b1,Db1b2,Db1c2],
    [Dc1a1,Db1c1,Ds,Dc1a2,Db2c1,Dc1c2],
    [Da1a2,Da2b1,Dc1a2,Ds,Da2b2,Dc2a2],
    [Da2b2,Db1b2,Db2c1,Da2b2,Ds,Db2c2],
    [Dc2a1,Db1c2,Dc1c2,Dc2a2,Db2c2,Ds]
])
Lp = u0 * np.log(De/L)

# a) Calculate Z' with 6x6 Method
Zp = Rp + 1j*frq*Lp
print("\na) Z-per-unit Matrix:")
print(table(np.asarray(np.around(Zp,4),dtype=str),tablefmt="fancy_grid"))

# Decompose and Calculate Equvalent Matrix
Za = Zp[:3,:3]
Zb = Zp[:3,3:6]
Zc = Zp[3:6,:3]
Zd = Zp[3:6,3:6]
Zbnew = Zb-Za
Zcnew = Zc-Za
Zdnew = Za - Zb - Zc + Zd
Zeq6 = Za - np.dot(Zbnew,np.dot(np.linalg.inv(Zdnew),Zcnew))
print("\nZ-per-unit Equivalent Matrix:")
print(table(np.asarray(np.around(Zeq6,4),dtype=str),tablefmt="fancy_grid"))

# Define GMD Calculator
def gmd(Ds,*args):
    # Find the Root from Number of Arguments
    root = len(args) + 1
    # Calculate the Root Term
    gmdx = Ds
    for dist in args:
        gmdx *= dist
    # Apply Root Calculation
    GMD = gmdx**(1/root)
    return(GMD)

# b) Calculate Z' from 3x3 Method
# Calculate GMD
GMD = gmd(GMR4,Da1b1,Dc1a1)
print("\nb) New GMD:",GMD)
# Calculate Equivalent Rself
Rselfeq = Rac4/3 + Rd
print("Equivalent Rself:",Rselfeq,"Ω")

# Calculate Zperlen
Zperlen = ep.zperlength(Rd=Rd,Rself=Rselfeq,Dab=Da1b1,Dbc=Db1c1,Dca=Dc1a1,Ds=GMD)
print("\nZ-per-unit Matrix:")
print(table(np.asarray(np.around(Zperlen,6),dtype=str),tablefmt="fancy_grid"))

```

```python
# Calculate Error
err = Zeq6 - Zperlen
print("\nError Matrix:")
print(table(np.asarray(np.around(err,6),dtype=str),tablefmt="fancy_grid"))

# c) Sequence Impedance Comparison
Zline4 = LineLen * Zperlen
print("\nc) Zline:")
print(table(np.asarray(np.around(Zline2,6),dtype=str),tablefmt="fancy_grid"))

# Find Zsequence
Zseq4 = ep.sequencez(Zline2)
print("\nZsequence:")
print(table(np.asarray(np.around(Zseq,6),dtype=str),tablefmt="fancy_grid"))

errZline = Zline4 - Zline2
errZseq = Zseq4 - Zseq
print("\nError Zline:")
print(table(np.asarray(np.around(errZline,6),dtype=str),tablefmt="fancy_grid"))
print("\nError Zsequence:")
print(table(np.asarray(np.around(errZseq,6),dtype=str),tablefmt="fancy_grid"))
```

```
Rd: 5.9214e-05 Ω        Rself: 0.560059214 Ω

a) Z-per-unit Matrix:
```

(0.5601+0.0009j)	(0.0001+0.0004j)	(0.0001+0.0004j)	(0.0001+0.0006j)	(0.0001+0.0004j)	(0.0001+0.0004j)
(0.0001+0.0004j)	(0.5601+0.0009j)	(0.0001+0.0004j)	(0.0001+0.0004j)	(0.0001+0.0006j)	(0.0001+0.0004j)
(0.0001+0.0004j)	(0.0001+0.0004j)	(0.5601+0.0009j)	(0.0001+0.0004j)	(0.0001+0.0004j)	(0.0001+0.0006j)
(0.0001+0.0006j)	(0.0001+0.0004j)	(0.0001+0.0004j)	(0.5601+0.0009j)	(0.0001+0.0004j)	(0.0001+0.0004j)
(0.0001+0.0004j)	(0.0001+0.0006j)	(0.0001+0.0004j)	(0.0001+0.0004j)	(0.5601+0.0009j)	(0.0001+0.0004j)
(0.0001+0.0004j)	(0.0001+0.0004j)	(0.0001+0.0006j)	(0.0001+0.0004j)	(0.0001+0.0004j)	(0.5601+0.0009j)

```
Z-per-unit Equivalent Matrix:
```

(0.2801+0.0008j)	(0.0001+0.0004j)	(0.0001+0.0004j)
(0.0001+0.0004j)	(0.2801+0.0008j)	(0.0001+0.0004j)
(0.0001+0.0004j)	(0.0001+0.0004j)	(0.2801+0.0008j)

```
b) New GMD: 1.0512193247270691
Equivalent Rself: 0.18672588066666668 Ω

Z-per-unit Matrix:
```

(0.186726+0.000594j)	(5.9e-05+0.000424j)	(5.9e-05+0.000439j)
(5.9e-05+0.000424j)	(0.186726+0.000594j)	(5.9e-05+0.000414j)
(5.9e-05+0.000439j)	(5.9e-05+0.000414j)	(0.186726+0.000594j)

```
Error Matrix:
```

(0.093333+0.000165j)	0j	(-0+0j)
2e-06j	(0.093333+0.000165j)	0j
(-0+0j)	0j	(0.093333+0.000165j)

```
c) Zline:
```

(19.464145+0.061469j)	(0.004145+0.029685j)	(0.004145+0.030736j)
(0.004145+0.029685j)	(19.464145+0.061469j)	(0.004145+0.028961j)
(0.004145+0.030736j)	(0.004145+0.028961j)	(19.464145+0.061469j)

```
Zsequence:
```

(19.472+0.121j)	(-0+0j)	0j

0j	(19.46+0.032j)	(0.001-0.001j)
(-0+0j)	(-0.001-0.001j)	(19.46+0.032j)

Error Zline:

(-6.393333-0.019861j)	0j	0j
0j	(-6.393333-0.019861j)	0j
0j	0j	(-6.393333-0.019861j)

Error Zsequence:

0j	0j	0j
0j	0j	0j
0j	0j	0j

5. Consider an untransposed line described in problem 2. Let the ground wire be 1/0 ACSR and recalculate the phase impedance matrix Zabc, the sequence impedance matrix Z012, and the unbalance factors. Compare with previous results from problem 2 for the same line without the ground wire. Assume phase conductors have same sag as problem 1, and that the ground wire is 7 feet lower at mid span than at the tower.

- Ground wire data:

  $Rac\_gw := 0.888 \frac{ohm}{mi}$ at 25C and 60Hz

  $GMR\_gw := 0.01113ft$ $diameter\_gw := .398in$

- Phase conductors same as in problem 2.

In [8]:

```
# Define Given Parameters
Rac_gw = 0.888
GMR_gw = 0.01113
dia_gw = 0.398/12 # feet

# Calculate Distances from Phase to Ground Conductors
Dag = np.sqrt(4**2+6.5**2)
Dbg = np.sqrt(5**2+11**2)
Dcg = np.sqrt(6**2+(8+6.5)**2)

# Calculate Zabc Matrix
Zabcpl = ep.zperlength(Rd=Rd,Rself=Rself,Rgwac=Rac_gw,
                       De=De,Dab=Dab,Dbc=Dbc,Dca=Dca,Ds=GMR,
                       Dsgw=GMR_gw,Dagw=Dag,Dbgw=Dbg,Dcgw=Dcg,resolve=True)
Zabc = Zabcpl * LineLen
print("\nZabc Matrix:")
print(table(np.asarray(np.around(Zabc,6),dtype=str),tablefmt="fancy_grid"))

# Calculate Variance
print("\nZabc Matrix Variance (Ground Wire/No Ground Wire):")
print(table(np.asarray(np.around(Zabc-Zline2,6),dtype=str),tablefmt="fancy_grid"))
```

Zabc Matrix:

(39.20416+0.061464j)	(0.004159+0.029681j)	(0.004158+0.030732j)
(0.004159+0.029681j)	(39.204158+0.061465j)	(0.004157+0.028957j)
(0.004158+0.030732j)	(0.004157+0.028957j)	(39.204157+0.061465j)

Zabc Matrix Variance (Ground Wire/No Ground Wire):

(19.740015-4e-06j)	(1.4e-05-4e-06j)	(1.3e-05-4e-06j)
(1.4e-05-4e-06j)	(19.740013-4e-06j)	(1.2e-05-4e-06j)
(1.3e-05-4e-06j)	(1.2e-05-4e-06j)	(19.740012-4e-06j)

6.

Repeat problem 5 with the transposition of problem 3, part (d).

In [ ]:

```

```

7.

Repeat problems 1-6 to calculate capacitances (only do 4(b) not (a) and (c)).

In [ ]:

```

```

In [ ]:

```

```