Joe Stanley

ECE 524 - HWK 6 Calculations

In [1]:
```
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import eepower as eep
from eepower import p,n,u,m,k,M
```

In [10]:
```
# Part B

# Evaluate Source Voltage (in RMS)
print(500*k/np.sqrt(2))
# Evaluate Source Impedance
Zsrc = eep.zsource(50000*M,500*k,12,perunit=False)
print("Source Impedance:",np.round(Zsrc,6),"Ω\tInductance:",np.round(eep.rea
```

```
353553.390593
Source Impedance: (0.415227+4.982729j) Ω        Inductance: 13.2171 mH
```

In [23]:

```python
# Part C

# Define Trans. Line Charging Current Calculation
def transcharge(length,C=None,Dab=0,Dbc=0,Dca=0,radius=0,
                VLN=None,VLL=None,freq=60):
    """
    transcharge Function

    Evaluates the charging current necessary to charge
    a three-phase transmission system characterized by
    the capacitance (C) or equivalently, the distances
    between each of the conductors and the radius of
    the conductors along with line length and voltage.

    Parameters
    ----------
    length:     float
                The line length in miles.
    C:          float, optional
                Shunt capacitance in Farads, optionally
                may be ignored if Dab, Dbc, Dca, and
                radius are set appropriately.
    Dab:        float, optional
                Distance from conductor A to B in feet.
    Dbc:        float, optional
                Distance from conductor B to C in feet.
    Dca:        float, optional
                Distance from conductor C to A in feet.
    radius:     float, optional
                Conductor cable radius in inches.
    VLN:        float, optional
                Line-to-Neutral voltage magnitude
    VLL:        float, optional
                Line-to-Line voltage magnitude
    freq:       float, optional
                System frequency in Hz, default=60

    Returns
    -------
    I:          float
                Charging current in amps per phase.
    """
    # Condition Inputs
    w = 2*np.pi*freq
    if C == None:
        DEQ = (Dab*Dbc*Dca)**(-1/3)
        C = 0.0388/(np.log10(DEQ/(radius/12)))
    if VLN != None:
        V = VLN
    if VLL != None:
        V = VLL/np.sqrt(3)
    # Calculate
    I = w*C*length*V
    return(I)

# Define C/L Calculator
```

```
def CL(v,Z):
    C = 1/(v*Z)
    L = C*Z**2
    return(C,L)

# From ATP HWK6.lib File --- C:\ProgramData\ATP\Atpdraw\ATP\HWK6.lib
# KARD  3  3  4  4  5  5
# KARG  1  4  2  5  3  6
# KBEG  3  9  3  9  3  9
# KEND  8 14  8 14  8 14
# KTEX  1  1  1  1  1  1
# /BRANCH
# $VINTAGE, 1
# -1IN___AOUT__A             5.07282E-01 5.52872E+02 1.22069E+05 1.50000E+02
# -2IN___BOUT__B             8.13006E-03 2.47178E+02 1.81727E+05 1.50000E+02
# -3IN___COUT__C
# $VINTAGE, -1,
# $EOF
# ARG, IN___A, IN___B, IN___C, OUT__A, OUT__B, OUT__C
R0 = 5.07282E-01
R1 = 8.13006E-03
R2 = R1
Zc0 = 5.52872E+02
Zc1 = 2.47178E+02
Zc2 = Zc1
v0 = 1.22069E+05
v1 = 1.81727E+05
C0,L0 = CL(v0,Zc0)
C1,L1 = CL(v1,Zc1)
C2,L2 = C1,L1
# Evaluate Sequence Impedances
Z0 = R0 + eep.phasorz(L=L0)
Z1 = R1 + eep.phasorz(L=L1)
print("Zero Sequence Impedance:",np.round(Z0,5),"Ω/mi\tCapacitance:",round(C
print("Positive Sequence Impedance:",np.round(Z1,5),"Ω/mi\tCapacitance:",rou
print("Negative Sequence Impedance:",np.round(Z1,5),"Ω/mi\tCapacitance:",rou
# Calculate Pos. Seq. Charging Current
Ichg = transcharge(150,C1,VLL=500*k)
print("Positive Sequence Charging Current:",Ichg,"A")
```

```
Zero Sequence Impedance: (0.50728+1.70746j) Ω/mi        Capacitance: 14.81733 n
F
Positive Sequence Impedance: (0.00813+0.51277j) Ω/mi    Capacitance: 22.26234 n
F
Negative Sequence Impedance: (0.00813+0.51277j) Ω/mi    Capacitance: 22.26234 n
F
Positive Sequence Charging Current: 363.414694392 A
```

In [ ]:
```
1
```