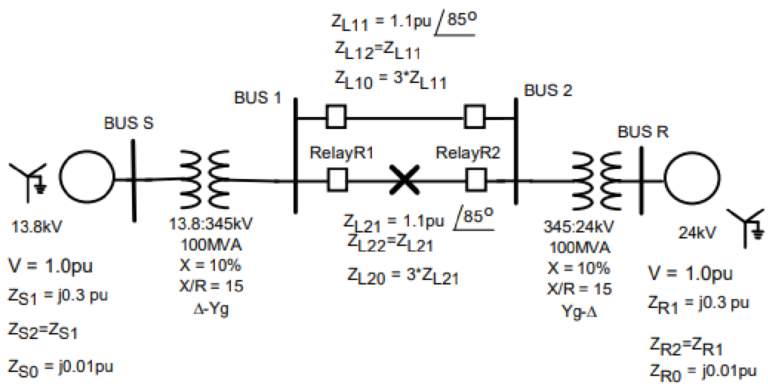
Joe Stanley

ECE 523 - HWK 4

In [1]:
```python
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import tabulate as tab
import electricpy as ep
from electricpy.constants import *
import numdifftools as ndt
```

1. Do the following for the circuit below using Z_{bus} matrix methods assuming faults 33% of the way down line 2 (the lower of the two lines). No change of base calculations needed.
   a. Set the voltage source at Bus S is 1.0 at -30 degrees (this is to account for the transformer phase shift), and the voltage at Bus R to be 1.0 is -50 degrees. Calculate the prefault voltage magnitude and angle at each bus, including the fault point based on the prefault power flow. Check your results with a Powerworld or a similar program.
   b. Calculate the voltages and currents in the sequence domain and in the abc domain at RelayR1 and RelayR2, for 3 phase, SLG, LL, and DLG faults with Rf = 0.3 pu (for the DLG put the resistance in the ground path). Again, check your results with Powerworld or a similar program.



In [16]:

```python
# Define Parameters
ZS1 = 0.3j
ZS0 = 0.01j
ZR1 = ZS1
ZR0 = ZS0
ZL11 = ep.phasor(1.1,85)
print(ZL11)
ZL10 = 3*ZL11
print(ZL10)
ZL21 = ZL11
ZL20 = 3*ZL21
ZT1 = ep.rxrecompose(0.1,15)
ZT2 = ZT1

# Generate Z-Bus Matricies
# 1 - 2 - f - S - R
ybus1 = np.array([
    [1/ZL11+1/(ZL21*33/100)+1/ZT1, -1/ZL11, -1/(ZL21*33/100), -en30/ZT1, 0],
    [-1/ZL11, 1/ZL11+1/(ZL21*67/100)+1/ZT2, -1/(ZL21*67/100), 0, -en30/ZT2],
    [-1/(ZL21*33/100), -1/(ZL21*67/100), 1/(ZL21*33/100)+1/(ZL21*67/100), 0, 0],
    [-e30/ZT1, 0, 0, 1/ZT1+1/ZS1, 0],
    [0, -e30/ZT2, 0, 0, 1/ZT2+1/ZR1]
])
zbus1 = np.linalg.inv(ybus1)
zbus2 = np.linalg.inv(np.array([
    [1/ZL11+1/(ZL21*33/100)+1/ZT1, -1/ZL11, -1/(ZL21*33/100), -e30/ZT1, 0],
    [-1/ZL11, 1/ZL11+1/(ZL21*67/100)+1/ZT2, -1/(ZL21*67/100), 0, -e30/ZT2],
    [-1/(ZL21*33/100), -1/(ZL21*67/100), 1/(ZL21*33/100)+1/(ZL21*67/100), 0, 0],
    [-en30/ZT1, 0, 0, 1/ZT1+1/ZS1, 0],
    [0, -en30/ZT2, 0, 0, 1/ZT2+1/ZR1]
]))
zbus0 = np.linalg.inv(np.array([
    [1/ZL10+1/(ZL20*33/100), -1/ZL10, -1/(ZL20*33/100), 0, 0],
    [-1/ZL10, 1/ZL10+1/(ZL20*67/100), -1/(ZL20*67/100), 0, 0],
    [-1/(ZL20*33/100), -1/(ZL20*67/100), 1/(ZL20*33/100)+1/(ZL20*67/100), 0, 0],
    [0, 0, 0, 1/ZS0, 0],
    [0, 0, 0, 0, 1/ZR0]
]))
def around(arr,val):
    tol = 1e+14
    arr = np.around(arr,val)
    arr.real[abs(arr.real) > tol] = 0.0
    arr.imag[abs(arr.imag) > tol] = 0.0
    return(arr)
print("\nPositive Sequence Z-Bus:")
print(tab.tabulate(np.asarray(np.around(zbus1,3),dtype=str),tablefmt="fancy_grid"))
print("\nNegative Sequence Z-Bus:")
print(tab.tabulate(np.asarray(np.around(zbus2,3),dtype=str),tablefmt="fancy_grid"))
print("\nZero Sequence Z-Bus:")
print(tab.tabulate(np.asarray(around(zbus0,5),dtype=str),tablefmt="fancy_grid"))

# Calculate Basic Power Transfer from Sending to Receiving
Vsnd = ep.phasor(1,-30)
Vrec = ep.phasor(1,-50)
Zsr1 = zbus1[3][4]
print("Send-Receive Positive Sequence Impedance:",np.around(Zsr1,3),"Ω-pu")
Pflow = ep.powerflow(Vsnd,Vrec,Zsr1)
print("Power Flow:",Pflow,"pu-VA")

# Define Lists for Power Flow Analysis
Vlist = [[None,None],[None,None],[None,None],[1,-np.radians(30)],[None,None]]
Plist = [0,0,0,None,-round(Pflow.real,3)]
Qlist = [0,0,0,None,-round(Pflow.imag,3)]

# Peform Power Flow Calculation
res, ct = ep.sim.mbuspowerflow(ybus1,Vlist,Plist,Qlist,
                               degrees=True,split=True,returnct=True,slackbus=3)
ang, mag = res
print(ang,mag)
print(ct)
```

```
(0.09587131702242396+1.0958141679009201j)
(0.2876139510672719+3.2874425037027604j)

Positive Sequence Z-Bus:
╒════════════════╤═════════════════╤════════════════╤════════════════╤════════════════╕
│ (0.008+0.281j) │ (-0.001+0.119j) │ (0.005+0.228j) │ (0.108+0.182j) │ (0.042+0.078j) │
```

(-0.001+0.119j)	(0.008+0.281j)	(0.002+0.172j)	(0.042+0.078j)	(0.108+0.182j)
(0.005+0.228j)	(0.002+0.172j)	(0.025+0.452j)	(0.086+0.147j)	(0.064+0.112j)
(-0.103+0.184j)	(-0.047+0.076j)	(-0.085+0.148j)	(0.003+0.233j)	(-0.003+0.067j)
(-0.047+0.076j)	(-0.103+0.184j)	(-0.065+0.111j)	(-0.003+0.067j)	(0.003+0.233j)

Negative Sequence Z-Bus:

(0.008+0.281j)	(-0.001+0.119j)	(0.005+0.228j)	(-0.103+0.184j)	(-0.047+0.076j)
(-0.001+0.119j)	(0.008+0.281j)	(0.002+0.172j)	(-0.047+0.076j)	(-0.103+0.184j)
(0.005+0.228j)	(0.002+0.172j)	(0.025+0.452j)	(-0.085+0.148j)	(-0.065+0.111j)
(0.108+0.182j)	(0.042+0.078j)	(0.086+0.147j)	(0.003+0.233j)	(-0.003+0.067j)
(0.042+0.078j)	(0.108+0.182j)	(0.064+0.112j)	(-0.003+0.067j)	(0.003+0.233j)

Zero Sequence Z-Bus:

0j	0j	0j	0j	0j
0j	0j	0j	0j	0j
0j	0j	0j	0j	0j
0j	0j	0j	0.01j	0j
0j	0j	0j	0j	0.01j

```
Send-Receive Positive Sequence Impedance: (-0.003+0.067j) Ω-pu
Power Flow: (-0.23283588191211826-5.12234860047134j) pu-VA
```

```
C:\Users\Joe Stanley\Anaconda3\lib\site-packages\electricpy-0.0.6-py3.6.egg\electricpy\sim.py:806: UserWarning: W
ARNING: Singular matrix, attempting LSQ method.
```

```
[-1.02277047e+12  3.55713241e+01  8.42739959e-01 -3.45680717e+00] [-1.10405705e-10  1.00000000e+00  1.00000000e+0
0  1.00000000e+00]
7
```

2. Given the system below do the following:

(a) Sketch the sequence equivalent circuits series faults occuring on line 1 (Bus 2-Bus3) and reduce them to simplified equivalents. The voltages given are at BUS1 and BUS4, ***not the voltages behind the source impedances.***

(b) Determine the phase currents from Bus1 to Bus 5 and from Bus 2 to Bus 3 if phase C is open on line 1 (treat it as the breakers at Bus 2 on phase C is open and phases A and B are closed)

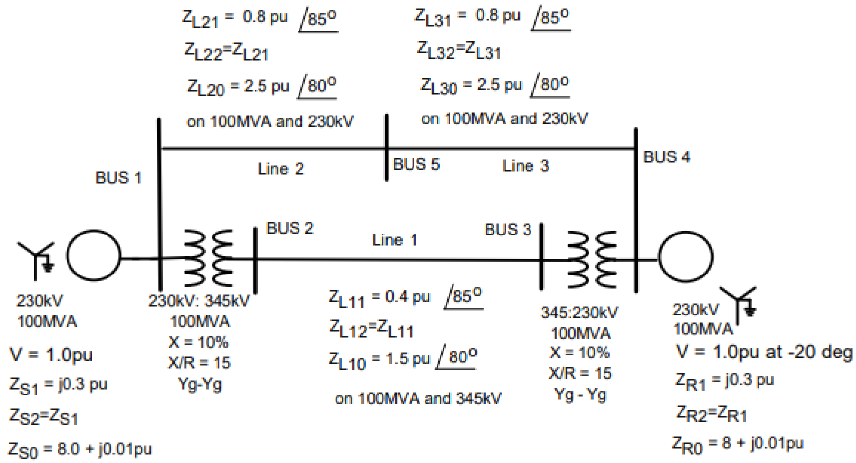
(c) Repeat part (b) if the transformer from Bus 1-Bus 2 is Y ungrounded on the HV side.

(d) Determine the phase currents from Bus1 to Bus 5 and from Bus 2 to Bus 3 if phases B and C are open on line 1(treat this as the breaker having phase A closed and phases B and C open)

(e) Repeat part (d) if the transformer from Bus 1-Bus 2 is Y ungrounded on the HV side.

(f) Suppose that line 3 is series compensated, with each phase having a capacitive impedance of -j0.3pu. The capacitor on phase A is bypassed by a misoperating circuit breaker and phases B and C are inserted. Calculate the currents from Bus1 to Bus 5 and from Bus 2 to Bus 3. Model Za and Zb as being the capacitor only. Do not lump in the line impedance.

(g) Verify your results using transient simulation



Z_L21 = 0.8 pu /85°
Z_L22=Z_L21
Z_L20 = 2.5 pu /80°
on 100MVA and 230kV
Z_L31 = 0.8 pu /85°
Z_L32=Z_L31
Z_L30 = 2.5 pu /80°
on 100MVA and 230kV
BUS 1
Line 2
BUS 5
Line 3
BUS 4
BUS 2
Line 1
BUS 3
230kV
100MVA
V = 1.0pu
Z_S1 = j0.3 pu
Z_S2=Z_S1
Z_S0 = 8.0 + j0.01pu
230kV: 345kV
100MVA
X = 10%
X/R = 15
Yg-Yg
Z_L11 = 0.4 pu /85°
Z_L12=Z_L11
Z_L10 = 1.5 pu /80°
on 100MVA and 345kV
345:230kV
100MVA
X = 10%
X/R = 15
Yg - Yg
230kV
100MVA
V = 1.0pu at -20 deg
Z_R1 = j0.3 pu
Z_R2=Z_R1
Z_R0 = 8 + j0.01pu

In [19]:

```python
# Define Givens
VR = ep.phasor(1,-20)
VS = 1 # no known angle
Zs1 = 0.3j
Zs0 = 8+0.01j
Zr1 = 0.3j
Zr0 = 8+0.01j
Zt1 = ep.rxrecompose(0.1,15)
Zt2 = ep.rxrecompose(0.1,15)
ZL11 = ep.phasor(0.4,85)
ZL10 = ep.phasor(1.5,80)
ZL21 = ep.phasor(0.8,85)
ZL20 = ep.phasor(2.5,80)
ZL31 = ep.phasor(0.8,85)
ZL30 = ep.phasor(2.5,80)

# Define Positive Sequence Y-Bus Matrix
# BUS: 1  -  2  -  3  -  4  -  5
ybus1 = np.array([
    [1/Zs1+1/ZL21+1/Zt1,-1/Zt1,0,0,-1/ZL21],
    [-1/Zt1,1/Zt1+1/ZL11,-1/ZL11,0,0],
    [0,-1/ZL11,1/ZL11+1/Zt2,-1/Zt2,0],
    [0,0,-1/Zt2,1/Zt2+1/ZL31+1/Zr1,-1/ZL31],
    [-1/ZL21,0,0,-1/ZL31,1/ZL21+1/ZL31],
])
# Define Zero Sequence Y-Bus Matrix
ybus0 = np.array([
    [1/Zs0+1/ZL20+1/Zt1,-1/Zt1,0,0,-1/ZL20],
    [-1/Zt1,1/Zt1+1/ZL10,-1/ZL10,0,0],
    [0,-1/ZL10,1/ZL10+1/Zt2,-1/Zt2,0],
    [0,0,-1/Zt2,1/Zt2+1/ZL30+1/Zr0,-1/ZL30],
    [-1/ZL20,0,0,-1/ZL30,1/ZL20+1/ZL30],
])

# Calculate Z-Bus Matrices
zbus1 = np.linalg.inv(ybus1)
zbus2 = np.linalg.inv(ybus1)
zbus0 = np.linalg.inv(ybus0)
print("\nPositive Sequence Z-Bus:")
print(tab.tabulate(np.asarray(np.around(zbus1,3),dtype=str),tablefmt="fancy_grid"))
print("\nNegative Sequence Z-Bus:")
print(tab.tabulate(np.asarray(np.around(zbus2,3),dtype=str),tablefmt="fancy_grid"))
print("\nZero Sequence Z-Bus:")
print(tab.tabulate(np.asarray(np.around(zbus0,3),dtype=str),tablefmt="fancy_grid"))

# Evaluate the Voltage at Faulted Bus
Pflow = ep.powerflow(VR,VS,zbus1[1][4])
Pflow = 0 + Pflow.imag*1j # Real Part is Essentially Zero
print("\nPower Flow:",Pflow,"pu-VA")

# Evaluate Faults for 1-Pole-Open
Zseq = [zbus0[2][2], zbus1[2][2], zbus2[2][2]]
Iphs = ep.fault.poleopen1(1,Zseq,sequence=False,reference='C')
print("\nBus 2 Phase Currents (1-Pole Open Case):")
ep.cprint(Iphs,"A-pu",["IA","IB","IC"])

# Evaluate Fault for 2-Pole-Open
Zseq = [zbus0[2][2], zbus1[2][2], zbus2[2][2]]
Iphs = ep.fault.poleopen2(1,Zseq,sequence=False,reference='A')
print("\nBus 2 Phase Currents (2-Pole Open Case):")
ep.cprint(Iphs,"A-pu",["IA","IB","IC"])
```

Positive Sequence Z-Bus:

(0.003+0.213j)	(0.002+0.192j)	(-0.002+0.108j)	(-0.003+0.087j)	(-0+0.15j)
(0.002+0.192j)	(0.008+0.261j)	(-0.001+0.139j)	(-0.002+0.108j)	(-0+0.15j)
(-0.002+0.108j)	(-0.001+0.139j)	(0.008+0.261j)	(0.002+0.192j)	(-0+0.15j)
(-0.003+0.087j)	(-0.002+0.108j)	(0.002+0.192j)	(0.003+0.213j)	(-0+0.15j)
(-0+0.15j)	(-0+0.15j)	(-0+0.15j)	(-0+0.15j)	(0.035+0.548j)

Negative Sequence Z-Bus:

(0.003+0.213j)	(0.002+0.192j)	(-0.002+0.108j)	(-0.003+0.087j)	(-0+0.15j)
(0.002+0.192j)	(0.008+0.261j)	(-0.001+0.139j)	(-0.002+0.108j)	(-0+0.15j)
(-0.002+0.108j)	(-0.001+0.139j)	(0.008+0.261j)	(0.002+0.192j)	(-0+0.15j)
(-0.003+0.087j)	(-0.002+0.108j)	(0.002+0.192j)	(0.003+0.213j)	(-0+0.15j)
(-0+0.15j)	(-0+0.15j)	(-0+0.15j)	(-0+0.15j)	(0.035+0.548j)

Zero Sequence Z-Bus:

(4.075+0.308j)	(4.069+0.272j)	(3.931-0.262j)	(3.925-0.298j)	(4+0.005j)
(4.069+0.272j)	(4.071+0.334j)	(3.936-0.224j)	(3.931-0.262j)	(4+0.005j)
(3.931-0.262j)	(3.936-0.224j)	(4.071+0.334j)	(4.069+0.272j)	(4+0.005j)
(3.925-0.298j)	(3.931-0.262j)	(4.069+0.272j)	(4.075+0.308j)	(4+0.005j)
(4+0.005j)	(4+0.005j)	(4+0.005j)	(4+0.005j)	(4.217+1.236j)

```
Power Flow: 2.280134288837792j pu-VA

Bus 2 Phase Currents (1-Pole Open Case):
[['IA 3.494 ∠ -178.779° A-pu']
 ['IB 3.132 ∠ 2.12° A-pu']
 ['IC 0.0 ∠ 0.0° A-pu']]

Bus 2 Phase Currents (2-Pole Open Case):
[['IA 0.719 ∠ -11.834° A-pu']
 ['IB 0.0 ∠ 90.0° A-pu']
 ['IC 0.0 ∠ 90.0° A-pu']]
```

In [4]:
```
1
```

```
(0.006666666666666667+0.1j)
```

In [ ]:
```
1
```