Joe Stanley

ECE523 - HWK2

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

Problem 1:

Determine the symmetrical components for the following currents and sketch a phasor diagram, based on

a) Phase "a" referenced components

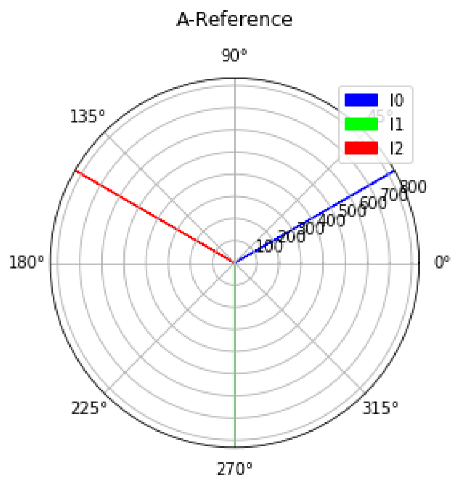
b) Phase "b" referenced components

c) Phase "c" referenced components

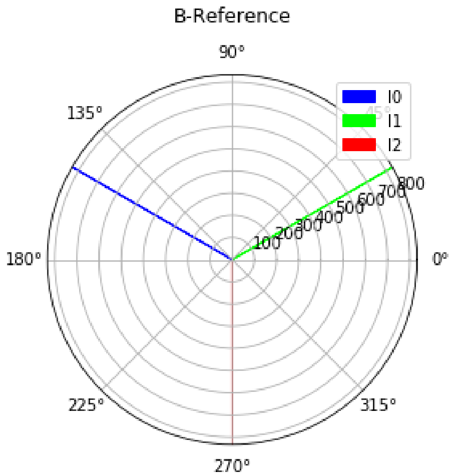
In [8]:

```python
# Define and Display Currents
currents = [[0,0],
            [0,-120],
            [2500,30]]
phs = ep.phasorlist(currents)
ep.cprint(phs,label=["IA","IB","IC"])

# a) "A"-Reference
symA = ep.abc_to_seq(phs)
ep.phasorplot(symA,legend=["I0","I1","I2"],title="A-Reference",
              colors=["#0000FF","#00FF00","#FF0000"])
ep.cprint(symA,label=["I0","I1","I2"])
print()

# b) "B"-Reference
symB = ep.abc_to_seq(phs,"B")
ep.phasorplot(symB,legend=["I0","I1","I2"],title="B-Reference",
              colors=["#0000FF","#00FF00","#FF0000"])
ep.cprint(symB,label=["I0","I1","I2"])
print()

# c) "C"-Reference
symC = ep.abc_to_seq(phs,"C")
ep.phasorplot(symC,legend=["I0","I1","I2"],title="C-Reference",
              colors=["#0000FF","#00FF00","#FF0000"])
ep.cprint(symC,label=["I0","I1","I2"])
print()
```

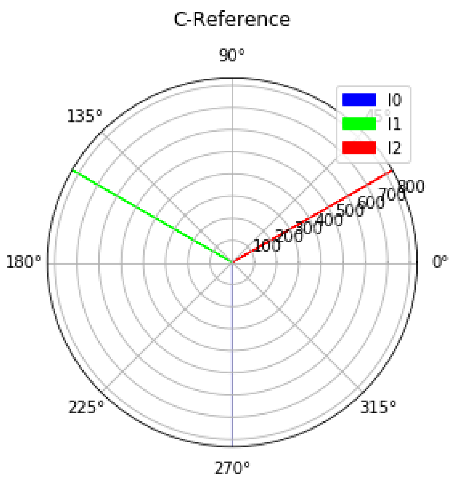
```
[['IA 0.0 ∠ 0.0°']
 ['IB 0.0 ∠ -180.0°']
 ['IC 2500.0 ∠ 30.0°']]
```




```
[['I0 833.333 ∠ 30.0°']
 ['I1 833.333 ∠ -90.0°']
 ['I2 833.333 ∠ 150.0°']]
```




```
[['I0 833.333 ∠ 150.0°']
 ['I1 833.333 ∠ 30.0°']
 ['I2 833.333 ∠ -90.0°']]
```




```
[['I0 833.333 ∠ -90.0°']
 ['I1 833.333 ∠ 150.0°']
 ['I2 833.333 ∠ 30.0°']]
```

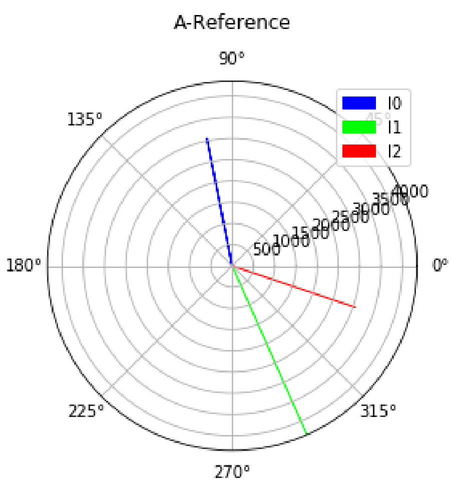
Problem 2:

Repeat problem 1. with the following currents:

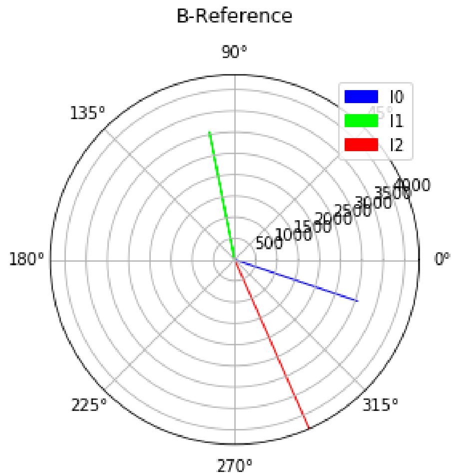
In [9]:

```
# Define and Display Currents
currents = [[4500,-25.84],
            [8503,-229.5],
            [4500,94.16]]
phs = ep.phasorlist(currents)
ep.cprint(phs,label=["IA","IB","IC"])

# a) "A"-Reference
symA = ep.abc_to_seq(phs)
ep.phasorplot(symA,legend=["I0","I1","I2"],title="A-Reference",
              colors=["#0000FF","#00FF00","#FF0000"])
ep.cprint(symA,label=["I0","I1","I2"])
print()

# b) "B"-Reference
symB = ep.abc_to_seq(phs,"B")
ep.phasorplot(symB,legend=["I0","I1","I2"],title="B-Reference",
              colors=["#0000FF","#00FF00","#FF0000"])
ep.cprint(symB,label=["I0","I1","I2"])
print()

# c) "C"-Reference
symC = ep.abc_to_seq(phs,"C")
ep.phasorplot(symC,legend=["I0","I1","I2"],title="C-Reference",
              colors=["#0000FF","#00FF00","#FF0000"])
ep.cprint(symC,label=["I0","I1","I2"])
print()
```

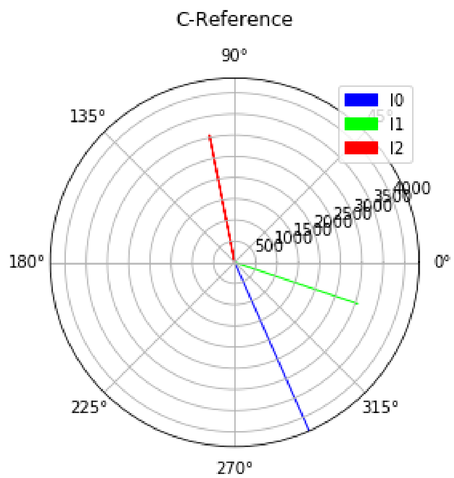
```
[['IA 4500.0 ∠ -25.84°']
 ['IB 8503.0 ∠ 130.5°']
 ['IC 4500.0 ∠ 94.16°']]
```




```
[['I0 3056.873 ∠ 101.311°']
 ['I1 4348.723 ∠ -66.214°']
 ['I2 3056.873 ∠ -18.689°']]
```




```
[['I0 3056.873 ∠ -18.689°']
 ['I1 3056.873 ∠ 101.311°']
 ['I2 4348.723 ∠ -66.214°']]
```




```
[['I0 4348.723 ∠ -66.214°']
 ['I1 3056.873 ∠ -18.689°']
 ['I2 3056.873 ∠ 101.311°']]
```

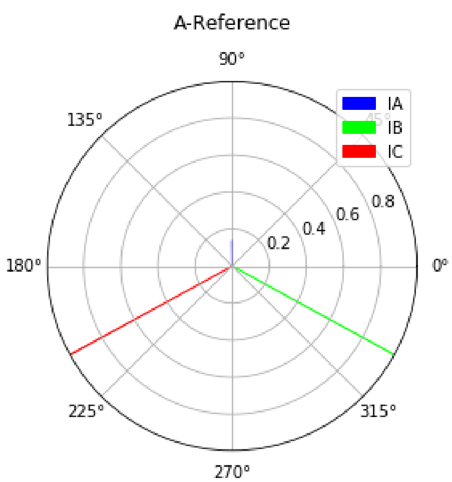
Problem 3:

Determine the phase voltage given the following phase "a" referenced symmetrical components. Repeat assuming they are instead phase "b" and then phase "c" referenced symmetrical components.

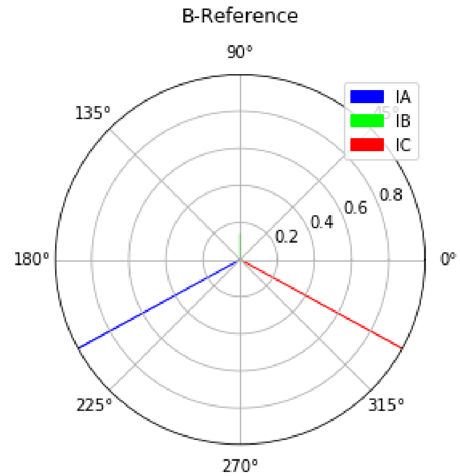
In [10]:

```python
# Define Symmetrical Voltages
volt = [[0.274,-90],
        [0.709,90],
        [0.299,-90]]
seq = ep.phasorlist(volt)
ep.cprint(seq,label=["I0","I1","I2"])

# a) "A"-Reference
phsA = ep.seq_to_abc(seq)
ep.phasorplot(phsA,legend=["IA","IB","IC"],title="A-Reference",
             colors=["#0000FF","#00FF00","#FF0000"])
ep.cprint(phsA,label=["IA","IB","IC"])
print()

# b) "B"-Reference
phsB = ep.seq_to_abc(seq,"B")
ep.phasorplot(phsB,legend=["IA","IB","IC"],title="B-Reference",
             colors=["#0000FF","#00FF00","#FF0000"])
ep.cprint(phsB,label=["IA","IB","IC"])
print()

# c) "C"-Reference
phsC = ep.seq_to_abc(seq,"C")
ep.phasorplot(phsC,legend=["IA","IB","IC"],title="C-Reference",
             colors=["#0000FF","#00FF00","#FF0000"])
ep.cprint(phsC,label=["IA","IB","IC"])
print()
```

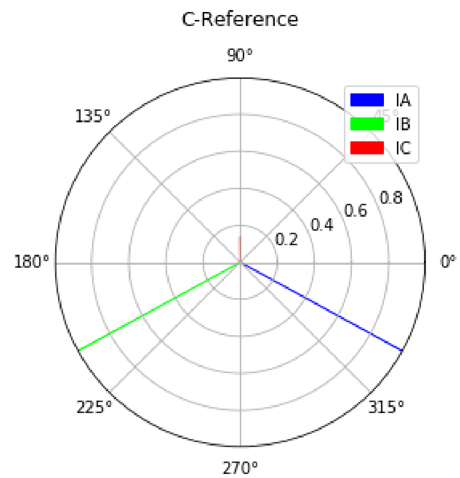
```
[['I0 0.274 ∠ -90.0°']
 ['I1 0.709 ∠ 90.0°']
 ['I2 0.299 ∠ -90.0°']]
```




```
[['IA 0.136 ∠ 90.0°']
 ['IB 0.996 ∠ -28.754°']
 ['IC 0.996 ∠ -151.246°']]
```




```
[['IA 0.996 ∠ -151.246°']
 ['IB 0.136 ∠ 90.0°']
 ['IC 0.996 ∠ -28.754°']]
```




```
[['IA 0.996 ∠ -28.754°']
 ['IB 0.996 ∠ -151.246°']
 ['IC 0.136 ∠ 90.0°']]
```

Problem 4: (text 2.10)

In [5]:

```python
# Create Arbitrary Impedance (Load)
Zaa = 0.5+0.1j
Zbb = 0.74+0.15j
Zcc = 0.89+0.11j
Zab = 0.1+0.6j
Zbc = 0.11+0.58j
Zca = 0.12+0.56j
Zabc = np.array([[Zaa,Zab,Zca],
                 [Zab,Zbb,Zbc],
                 [Zca,Zbc,Zcc]])

# Evaluate Sequence Impedance
Z012 = ep.sequencez(Zabc)
print("Sequence Impedance:\n",Z012)

# Create Balanced Voltage Source
Vsrc = ep.phasorlist([[1,0],
                      [1,-120],
                      [1,120]])
V012 = ep.abc_to_seq(Vsrc)
print("\nSequence Voltages of Balanced Voltage Source:")
ep.cprint(V012,label=["V0","V1","V2"])

# Evaluate Sequence Currents
I012 = np.linalg.inv(Z012).dot(V012)
print("\nSequence Currents:")
ep.cprint(I012,label=["I0","I1","I2"])

# Evaluate Phase Currents
Iabc = ep.seq_to_abc(I012)
print("\nPhase Currents:")
ep.cprint(Iabc,label=["IA","IB","IC"])

# Determine Neutral Current
In = np.sum(Iabc)
print("\nNeutral Current:")
ep.cprint(In,label="In:")

# Determine 3*Io
I0_3 = 3*I012[0]
print("\nZero-Sequence Current:")
ep.cprint(I0_3,label="3*I0:")

# Demonstrate that Currents are Equal
ion = r"$$I_{0}-I_{n}="+ep.clatex(I0_3-In,double=True,predollar=False)
```

```
Sequence Impedance:
 [[ 0.930+1.28j  -0.082+0.039j -0.128-0.059j]
 [-0.128-0.059j  0.600-0.46j  -0.117+0.022j]
 [-0.082+0.039j -0.093-0.042j  0.600-0.46j ]]

Sequence Voltages of Balanced Voltage Source:
[['V0 0.0 ∠ 0.0°']
 ['V1 1.0 ∠ -0.0°']
 ['V2 0.0 ∠ 90.0°']]

Sequence Currents:
[['I0 0.073 ∠ -29.092°']
 ['I1 1.338 ∠ 38.622°']
 ['I2 0.177 ∠ 97.877°']]

Phase Currents:
[['IA 1.458 ∠ 41.934°']
 ['IB 1.472 ∠ -85.131°']
 ['IC 1.089 ∠ 159.259°']]

Neutral Current:
In: 0.219 ∠ -29.092°

Zero-Sequence Current:
3*I0: 0.219 ∠ -29.092°
```

This demonstrates that $I_n = I_0$ as shown below by their subtracted total:

$$I_0 - I_n = 0.0\angle - 17.103°$$

Problem 5: (text 2.13)

In [6]:

```
# Define Givens
Vab = ep.phasor(1,30)
Vac = ep.phasor(1,-30)
Vbc = ep.phasor(1,-90)
Vca = Vac * 1j**2

# Determine Line-to-Neutral Voltages
Van = ep.phaseline(VLL=Vab,complex=True)
Vbn = ep.phaseline(VLL=Vbc,complex=True)
Vcn = ep.phaseline(VLL=Vca,complex=True)

# Generate Sequence Voltages
V012 = ep.abc_to_seq([Van,Vbn,Vcn])
ep.cprint(V012,label=["V0","V1","V2"])

# Create LaTeX String
Vtex = r'$$V_{012}='+ep.clatex(V012,predollar=False,double=True)
```

```
[['V0 0.0 ∠ -104.036°']
 ['V1 0.577 ∠ -0.0°']
 ['V2 0.0 ∠ 15.945°']]
```

Line-to-Neutral Voltages:

$$V_{012} = \begin{bmatrix} 0.0\angle - 104.036° \\ 0.577\angle - 0.0° \\ 0.0\angle 15.945° \end{bmatrix}$$

Problem 6:

In [7]:

```python
# Define Givens
CTR_H = 500
CTR_L = 5
CTR = CTR_H/CTR_L
PHS_L = [[12,-87],
         [4,-120],
         [4,120]]
PHS_L = ep.phasorlist(PHS_L)

# a) Primary Currents
print("a) Primary Currents:")
PHS_H = ep.primary(PHS_L,CTR)
ep.cprint(PHS_H)

# b) Secondary Sequence Currents
print("\nb) Sequence Components:")
PHS_L_012 = ep.abc_to_seq(PHS_L)
ep.cprint(PHS_L_012,label=["I0","I1","I2"])

# c) Residual Current
print("\nc) Neutral Current:")
I_r = np.sum(PHS_L)
ep.cprint(I_r,label="Residual:")
ep.cprint(3*PHS_L_012[0],label="3*I0:\t ")

# d) Delta Currents
PHS_L_D = ep.phaseline(Iline=ep.secondary(PHS_L,CTR),complex=True)
PHS_L_D_012 = ep.abc_to_seq(PHS_L_D)
PHS_L_D_012[0] = 0 # Force Zero-Sequence to Zero
print("\nd) Sequence Currents:")
ep.cprint(PHS_L_D_012,label=["I0","I1","I2"])
```

```
a) Primary Currents:
[['1200.0 ∠ -87.0°']
 ['400.0 ∠ -120.0°']
 ['400.0 ∠ 120.0°']]

b) Sequence Components:
[['I0 4.15 ∠ -105.716°']
 ['I1 4.922 ∠ -54.247°']
 ['I2 4.15 ∠ -105.716°']]

c) Neutral Current:
Residual: 12.449 ∠ -105.716°
3*I0:     12.449 ∠ -105.716°

d) Sequence Currents:
[['I0 0.0 ∠ 0.0°']
 ['I1 0.028 ∠ -24.247°']
 ['I2 0.024 ∠ -75.716°']]
```

In [ ]:

```python

```