Joe Stanley

ECE522 - EXAM1

In [28]:
```
# Import Necessary Libraries
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve
import electricpy as ep
from electricpy.constants import *

# Set Boolean Control for Report Style
debug = False
```

Problem I:

Part A:

Given the parameters described below, find: $\text{slip}_{\text{rated}}$, $\omega_{r-\text{rated}}$, and $\Lambda_{dr_{\text{rated}}}$. Rated is defined to mean that the operating conditions are such that: $|V_{dqs}| = 1.0\text{pu}$, $T_{em} = 1.0\text{pu}$, and $\omega_{es} = 1.0\text{pu}$

For this, we know:

$$V_{dqs} = r_s I_{dqs} + j\omega_{es}\left(L_s I_{dqs} + \tfrac{3}{2} L_{sr} I_{dqr}\right)$$
$$0 = r_r I_{dqr} + j(\omega_{es} - \omega_r)\Lambda_{dqr}$$
$$\Lambda_{dqr} = \tfrac{3}{2} L_{sr} I_{dqs} + L_r I_{dqr}$$
$$T_{em} = \frac{3p}{4}\frac{\frac{3}{2}L_{sr}}{L_r}\text{Im}\left(\overline{\Lambda_{dqr}} I_{dqs}\right)$$

Additionally, we will need to use a few additional equations to solve this system.

$$L_s = L_{LS} + L_m$$
$$L_r = L_{Lr} + L_m$$
$$L_{sr} = \tfrac{2}{3} \cdot (L_r - L_{Lr}) \cdot a_t$$

As a final note, we will make the following assumptions:

$$p = 4 \qquad a_t = 2$$

Using an iterative solver (shown in code below), we can now find the terms as:

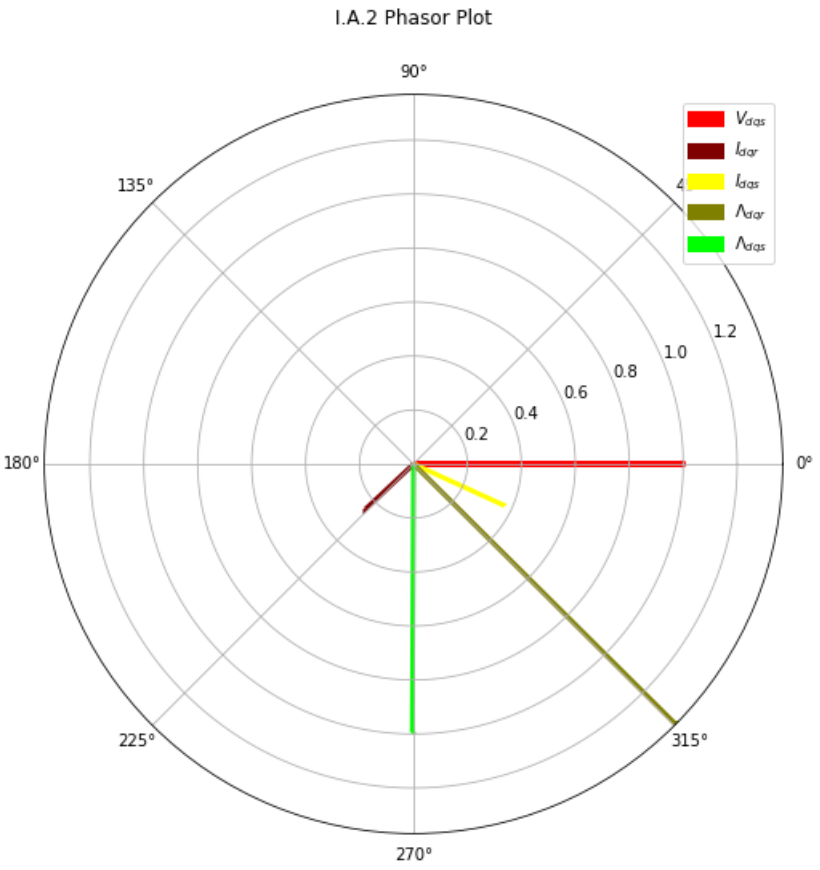
$$\begin{bmatrix} V_{\text{dqs}} \\ I_{\text{dqr}} \\ I_{\text{dqs}} \\ \Lambda_{\text{dqr}} \\ \Lambda_{\text{dqr}} \end{bmatrix} = \begin{bmatrix} 1.0\angle 0.0^\circ \\ 0.244\angle -136.784^\circ \\ 0.37\angle -24.54^\circ \\ 1.369\angle -44.783^\circ \\ 0.99\angle -90.266^\circ \end{bmatrix}$$

Sub-Part 1:

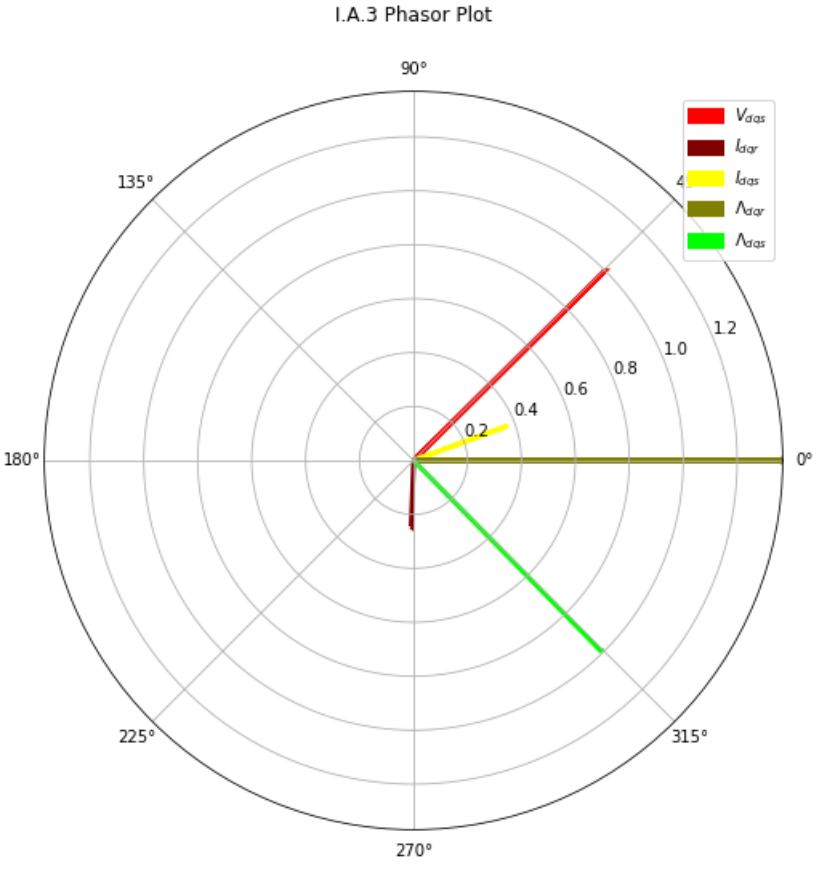
From simple manipulation and calculation, we can find the desired results to equal:

- $\text{slip}_{\text{rated}}$ =0.00532
- $\omega_{r_{\text{rated}}}$ =0.99468
- $\Lambda_{\text{dr}_{\text{rated}}}$ =1.36883

Sub-Part 2:


I.A.2 Phasor Plot
90°
135°
180°
225°
270°
315°
0°
0.2
0.4
0.6
0.8
1.0
1.2
V_{das}
I_{dar}
I_{das}
Λ_{dar}
Λ_{das}

Sub-Part 3:


I.A.3 Phasor Plot
90°
135°
180°
225°
270°
315°
0°
0.2
0.4
0.6
0.8
1.0
1.2
V_{das}
I_{dar}
I_{das}
Λ_{dar}
Λ_{das}

In [29]:

```
# Define Provided Machine Parameters
rs = 0.03 #pu
LLs = 0.1 #pu
Lm = 2.0 #pu
LLr = 0.1 #pu
rr = 0.03 #pu

# Define Rated Criteria
VdqsMag = 1
Tem = 1
wes = 1

# Assumptions
p = 4
at = 2

# Calculate Additional Inductance Terms
Ls = LLs + Lm
Lr = LLr + Lm
Lmr = 2/3 * (Lr - LLr)
Lsr = Lmr * at

# Define Equations Function as Solver
def equations(val):
    Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,wr = val
    A = (rs*Ids - wes*LAMqs) - VdqsMag
    B = rs*Iqs - wes*LAMds
    C = rr*Idr - (wes-wr)*LAMqr
    D = rr*Iqr + (wes-wr)*LAMdr
    E = (Ls*Ids + 3/2*Lsr*Idr) - LAMds
    F = (Ls*Iqs + 3/2*Lsr*Iqr) - LAMqs
    G = (3/2*Lsr*Ids+Lr*Idr) - LAMdr
    H = (3/2*Lsr*Iqs+Lr*Iqr) - LAMqr
    I = (3*p/4*(3/2*Lsr)/Lr*(LAMdr*Iqs-LAMqr*Ids)) - Tem
    return(A,B,C,D,E,F,G,H,I)

########################################################################
# A.1)

# Define Initial Guesses
Idr0 = -1
Iqr0 = -1
Ids0 = 1
Iqs0 = -1
LAMdr0 = 3/2*Lsr*Ids0 + Lr*Idr0
LAMqr0 = 3/2*Lsr*Iqs0 + Lr*Iqr0
LAMds0 = Ls*Ids0 + 3/2*Lsr*Idr0
LAMqs0 = Ls*Iqs0 + 3/2*Lsr*Iqr0
wr = 360

# Use Iterative Solver to Find Results
Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,wr = fsolve(equations,(
    Idr0,Iqr0,Ids0,Iqs0,LAMdr0,LAMqr0,LAMds0,LAMqs0,wr))

# Define Complex Composition Function
def complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs):
    # Return order: Vdqs, Idqr, Idqs, LAMdqr, LAMdqs
    return(np.array([
        Vds + 1j*Vqs,
        Idr + 1j*Iqr,
        Ids + 1j*Iqs,
        LAMdr+1j*LAMqr,
        LAMds+1j*LAMqs,
    ]))
Vds = VdqsMag
Vqs = 0
clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)

# Generate into Latex Vector
vect_desc = (r"$$\begin{bmatrix}V_{\text{dqs}}\\I_{\text{dqr}}\\"+
             r"I_{\text{dqs}}\\\Lambda_{\text{dqr}}\\ \Lambda_{\text{dqr}}"+
```

```
            "\end{bmatrix}=")
P1_latex = vect_desc + ep.clatex(clist,predollar=False,double=True)

# Calculate Desired Terms
s_rated = round((wes-wr)/wes,5)
w_rated = round(wr,5)
lamdr_rated = round(abs(LAMdqr),5)

########################################################################
# A.2)
# Use Complex Values to Plot Phasor Diagram
labels = [
    "$V_{dqs}$",
    "$I_{dqr}$",
    "$I_{dqs}$",
    "$\\Lambda_{dqr}$",
    "$\\Lambda_{dqs}$",
]
ep.phasorplot(clist,"I.A.2 Phasor Plot",labels,filename="I-A-2",size=8,linewidth=3,plot=debug)

########################################################################
# A.3)
# Calculate Phase Shift
shift = np.angle(LAMdqr,deg=True)
# Shift Phasors by Phase Shift Specified
clist *= ep.phs(-shift)
# Use Complex Values to Plot Phasor Diagram
ep.phasorplot(clist,"I.A.3 Phasor Plot",labels,filename="I-A-3",size=8,linewidth=3,plot=debug)
```

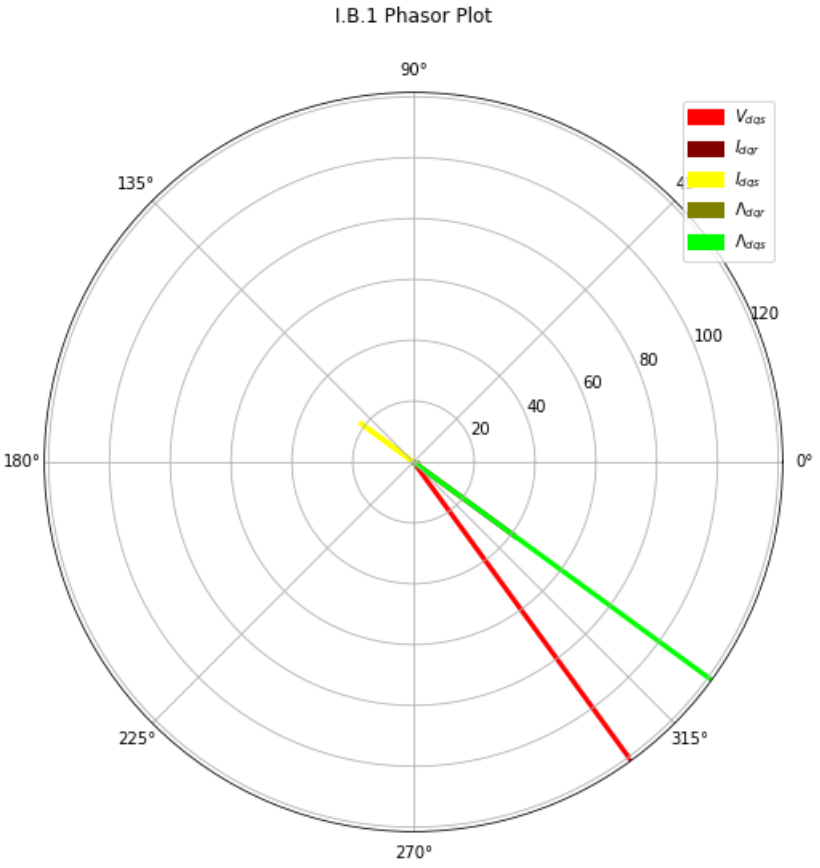
Part B:

1) $\omega_{rB1} := 0.0$ and $\Lambda_{drB1} = \Lambda_{dr\_rated}$

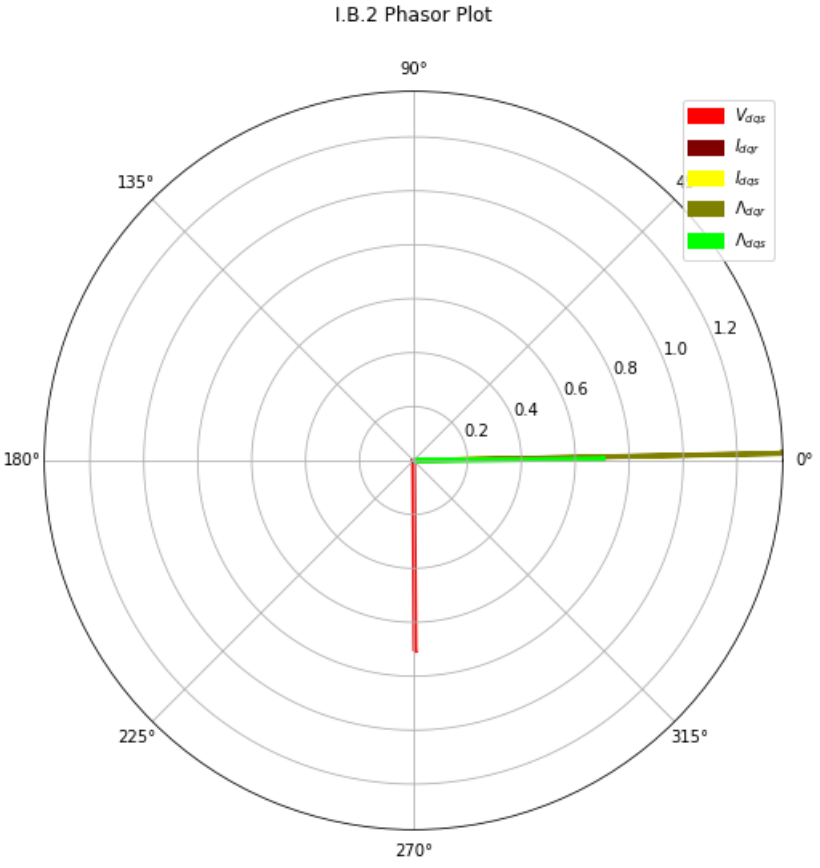
2) $\omega_{rB2} = \omega_{r\_rated}$ and $\Lambda_{drB2} = \Lambda_{dr\_rated}$

3) $\omega_{rB3} = 2 \cdot \omega_{r\_rated}$ and $\Lambda_{drB3} = \frac{\Lambda_{dr\_rated}}{2}$

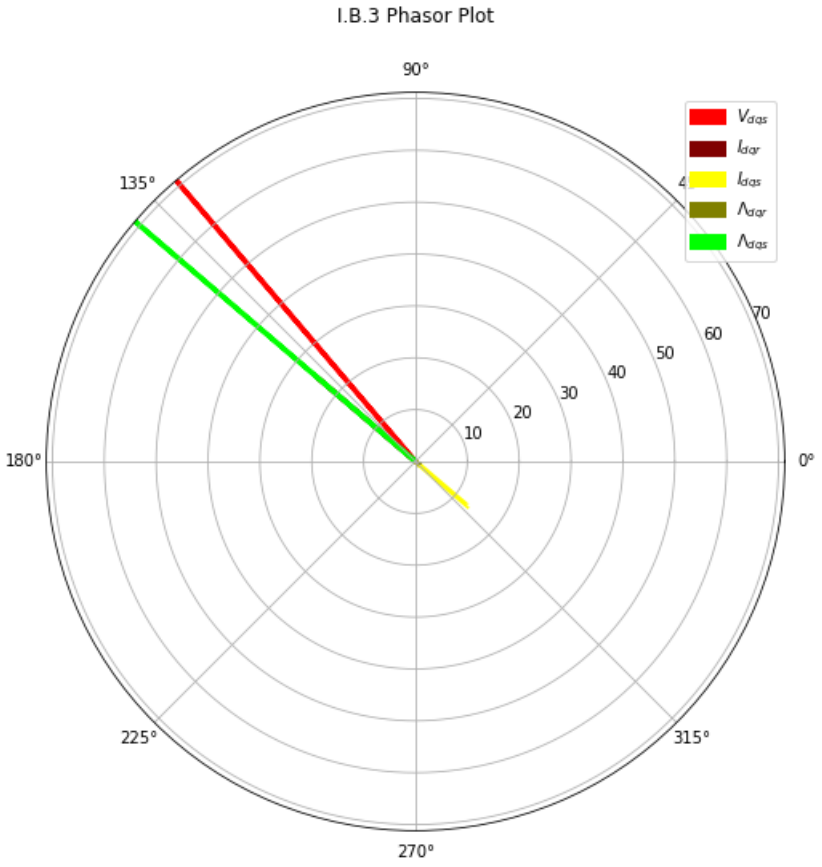
Sub-Part 1:


I.B.1 Phasor Plot
90°
135°
180°
225°
270°
315°
0°
20
40
60
80
100
120
V_{das}
I_{dar}
I_{das}
Λ_{dar}
Λ_{das}

Sub-Part 2:


I.B.2 Phasor Plot
90°
135°
180°
225°
270°
315°
0°
0.2
0.4
0.6
0.8
1.0
1.2
V_{das}
I_{dar}
I_{das}
Λ_{dar}
Λ_{das}

Sub-Part 3:


I.B.3 Phasor Plot
90°
135°
180°
225°
270°
315°
0°
10
20
30
40
50
60
70
V_{dqs}
I_{dqr}
I_{dqs}
Λ_{dqr}
Λ_{dqs}

In [30]:

```
# Re-Define Known Values
Tem = 0

# Define Equations Function as Solver
def equations_B(val,wr,LAMdr):
    Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = val
    A = (rs*Ids - wes*LAMqs) - Vds
    B = (rs*Iqs - wes*LAMds) - Vqs
    C = rr*Idr - (wes-wr)*LAMqr
    D = rr*Iqr + (wes-wr)*LAMdr
    E = (Ls*Ids + 3/2*Lsr*Idr) - LAMds
    F = (Ls*Iqs + 3/2*Lsr*Iqr) - LAMqs
    G = (3/2*Lsr*Ids+Lr*Idr) - LAMdr
    H = (3/2*Lsr*Iqs+Lr*Iqr) - LAMqr
    I = (3*p/4*(3/2*Lsr)/Lr*(LAMdr*Iqs-LAMqr*Ids)) - Tem
    return(A,B,C,D,E,F,G,H,I)

# Define Initial Guesses
Vds0 = 1
Vqs0 = 0
Idr0 = -1
Iqr0 = -1
Ids0 = 1
Iqs0 = -1
LAMqr0 = 3/2*Lsr*Iqs0 + Lr*Iqr0
LAMds0 = Ls*Ids0 + 3/2*Lsr*Idr0
LAMqs0 = Ls*Iqs0 + 3/2*Lsr*Iqr0

########################################################################
# B.1)
wr = 0.0
LAMdr = lamdr_rated
b1 = lambda x: equations_B(x,wr,LAMdr)

# Use Iterative Solver to Find Results
Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(b1,(
    Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))

# Generate Phasor Plot
clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
ep.phasorplot(clist,"I.B.1 Phasor Plot",labels,filename="I-B-1",size=8,linewidth=3,plot=debug)

########################################################################
# B.2)
wr = w_rated
LAMdr = lamdr_rated
b2 = lambda x: equations_B(x,wr,LAMdr)

# Use Iterative Solver to Find Results
Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(b2,(
    Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))

# Generate Phasor Plot
clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
ep.phasorplot(clist,"I.B.2 Phasor Plot",labels,filename="I-B-2",size=8,linewidth=3,plot=debug)

########################################################################
# B.3)
wr = 2*w_rated
LAMdr = lamdr_rated/2
b3 = lambda x: equations_B(x,wr,LAMdr)

# Use Iterative Solver to Find Results
Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(b3,(
    Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))

# Generate Phasor Plot
clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
ep.phasorplot(clist,"I.B.3 Phasor Plot",labels,filename="I-B-3",size=8,linewidth=3,plot=debug)
```

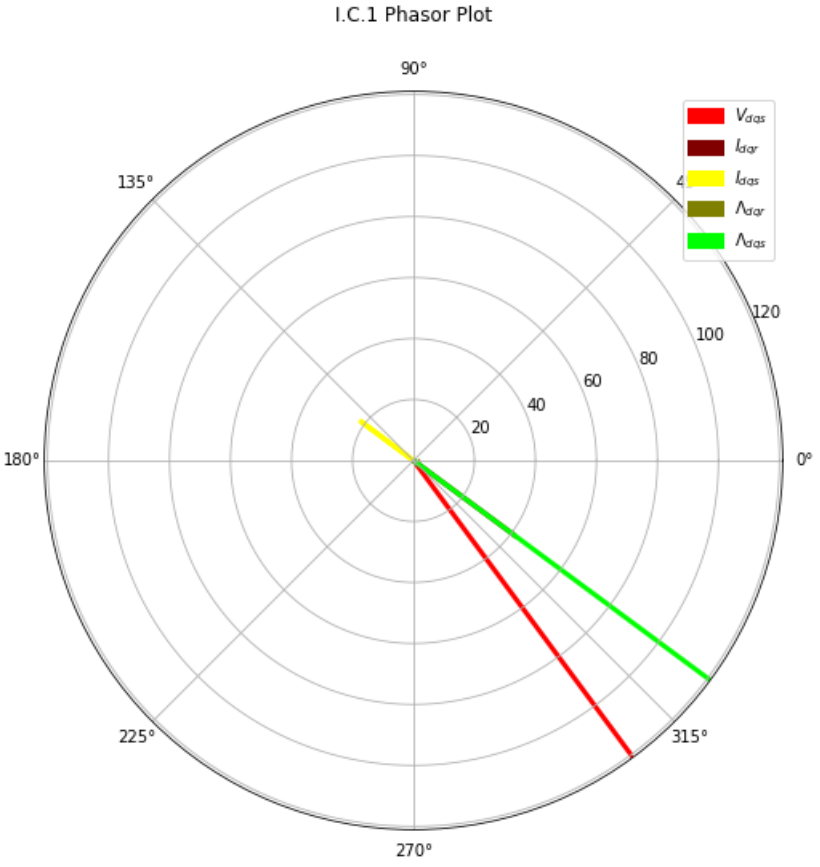
Part C:

1) $T_{EMC1} := 1.0 \cdot pu$ $\omega_{rC1} := 0.0$ and $\Lambda_{drC1} = \Lambda_{dr\_rated}$

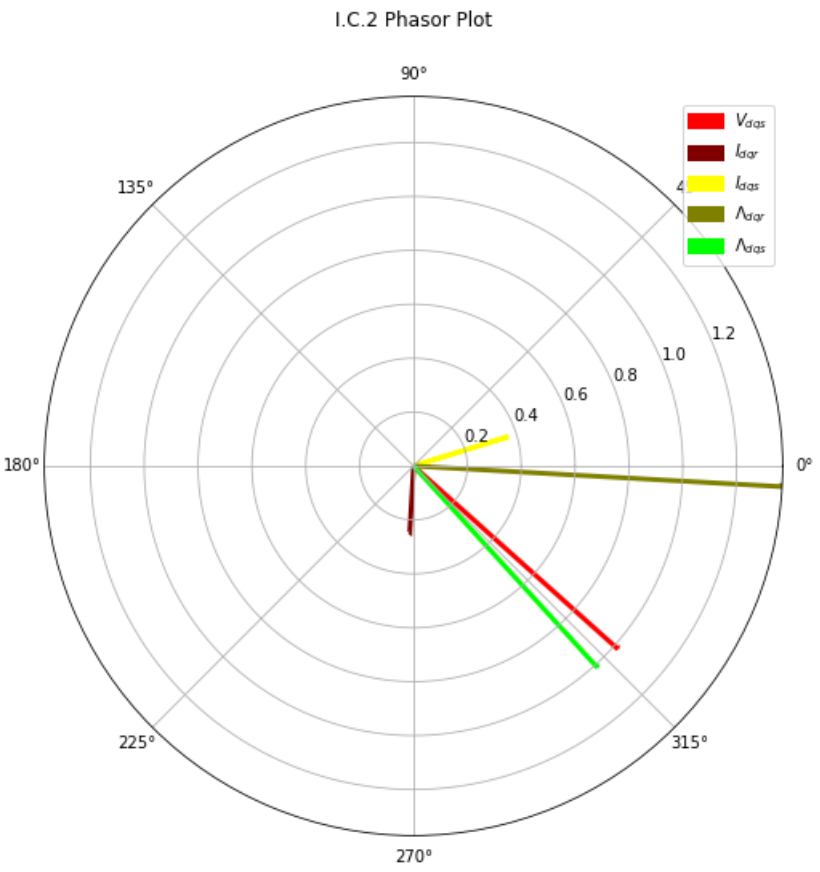
2) $T_{EMC2} := 1.0 \cdot pu$ $\omega_{rC2} = \omega_{r\_rated}$ and $\Lambda_{drC2} = \Lambda_{dr\_rated}$

3) $T_{EMC3} := 0.5 \cdot pu$ $\omega_{rC3} = 2 \cdot \omega_{r\_rated}$ and $\Lambda_{drC3} = \frac{\Lambda_{dr\_rated}}{2}$

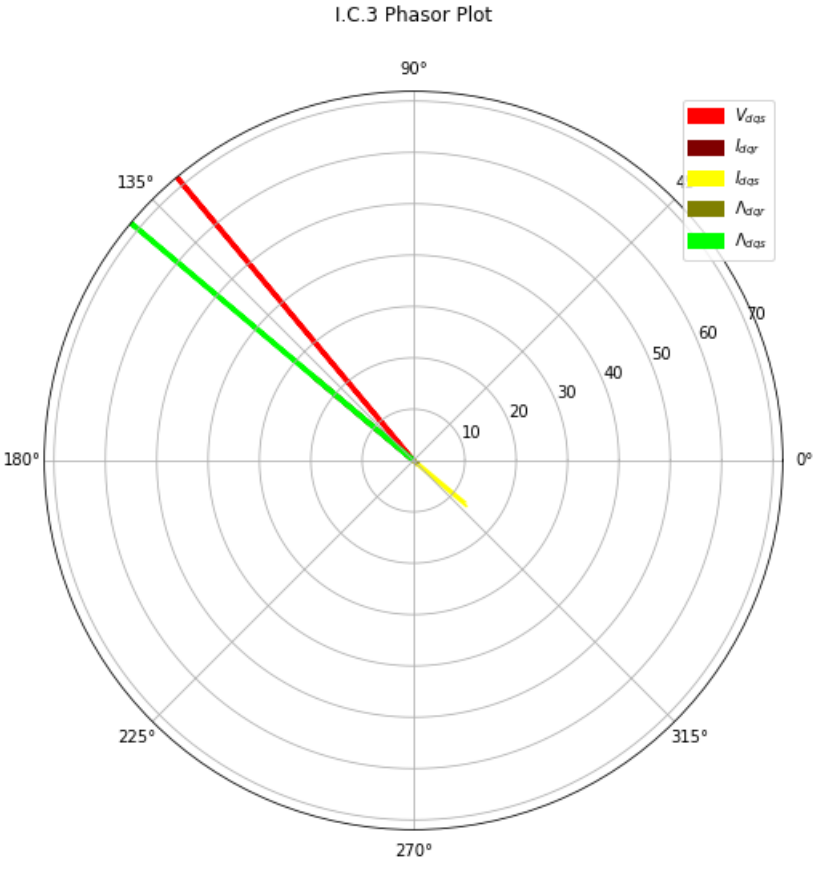
Sub-Part 1:



Sub-Part 2:


I.C.2 Phasor Plot
90°
135°
180°
225°
270°
315°
0°
0.2
0.4
0.6
0.8
1.0
1.2
V_{das}
I_{dar}
I_{das}
Λ_{dar}
Λ_{das}

Sub-Part 3:


I.C.3 Phasor Plot
90°
135°
180°
225°
270°
315°
0°
10
20
30
40
50
60
70
V_{das}
I_{dar}
I_{das}
Λ_{dar}
Λ_{das}

In [31]:

```
# Define Equations Function as Solver
def equations_C(val,wr,LAMdr,Tem):
    Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = val
    A = (rs*Ids - wes*LAMqs) - Vds
    B = (rs*Iqs - wes*LAMds) - Vqs
    C = rr*Idr - (wes-wr)*LAMqr
    D = rr*Iqr + (wes-wr)*LAMdr
    E = (Ls*Ids + 3/2*Lsr*Idr) - LAMds
    F = (Ls*Iqs + 3/2*Lsr*Iqr) - LAMqs
    G = (3/2*Lsr*Ids+Lr*Idr) - LAMdr
    H = (3/2*Lsr*Iqs+Lr*Iqr) - LAMqr
    I = (3*p/4*(3/2*Lsr)/Lr*(LAMdr*Iqs-LAMqr*Ids)) - Tem
    return(A,B,C,D,E,F,G,H,I)

# Define Initial Guesses
Vds0 = 1
Vqs0 = 0
Idr0 = -1
Iqr0 = -1
Ids0 = 1
Iqs0 = -1
LAMqr0 = 3/2*Lsr*Iqs0 + Lr*Iqr0
LAMds0 = Ls*Ids0 + 3/2*Lsr*Idr0
LAMqs0 = Ls*Iqs0 + 3/2*Lsr*Iqr0

########################################################################
# C.1)
Tem = 1.0
wr = 0.0
LAMdr = lamdr_rated
c = lambda x: equations_C(x,wr,LAMdr,Tem)

# Use Iterative Solver to Find Results
Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(c,(
    Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))

# Generate Phasor Plot
clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
ep.phasorplot(clist,"I.C.1 Phasor Plot",labels,filename="I-C-1",size=8,linewidth=3,plot=debug)

########################################################################
# C.2)
Tem = 1.0
wr = w_rated
LAMdr = lamdr_rated
c = lambda x: equations_C(x,wr,LAMdr,Tem)

# Use Iterative Solver to Find Results
Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(c,(
    Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))

# Generate Phasor Plot
clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
ep.phasorplot(clist,"I.C.2 Phasor Plot",labels,filename="I-C-2",size=8,linewidth=3,plot=debug)

########################################################################
# C.3)
Tem = 0.5
wr = 2*w_rated
LAMdr = lamdr_rated/2
c = lambda x: equations_C(x,wr,LAMdr,Tem)

# Use Iterative Solver to Find Results
Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(c,(
    Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))

# Generate Phasor Plot
clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
ep.phasorplot(clist,"I.C.3 Phasor Plot",labels,filename="I-C-3",size=8,linewidth=3,plot=debug)
```

In [ ]:

```
1
```