

Joe Stanley

ECE522 - EXAM1

```
In [28]: 1 # Import Necessary Libraries
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from scipy.optimize import fsolve
5 import electricpy as ep
6 from electricpy.constants import *
7
8 # Set Boolean Control for Report Style
9 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:

$$V_{dqs} = r_s I_{dqs} + j\omega_{es} \left(L_s I_{dqs} + \frac{3}{2} L_{sr} I_{dqr} \right)$$

$$0 = r_r I_{dqr} + j(\omega_{es} - \omega_r) \Lambda_{dqr}$$

$$\Lambda_{dqr} = \frac{3}{2} L_{sr} I_{dqs} + L_r I_{dqr}$$

$$T_{em} = \frac{3p}{4} \frac{\frac{3}{2} L_{sr}}{L_r} \text{Im} \left(\overline{\Lambda_{dqr}} I_{dqs} \right)$$

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

$$L_s = L_{LS} + L_m$$

$$L_r = L_{Lr} + L_m$$

$$L_{sr} = \frac{2}{3} \cdot (L_r - L_{Lr}) \cdot a_t$$

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

$$p = 4 \quad a_t = 2$$

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

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

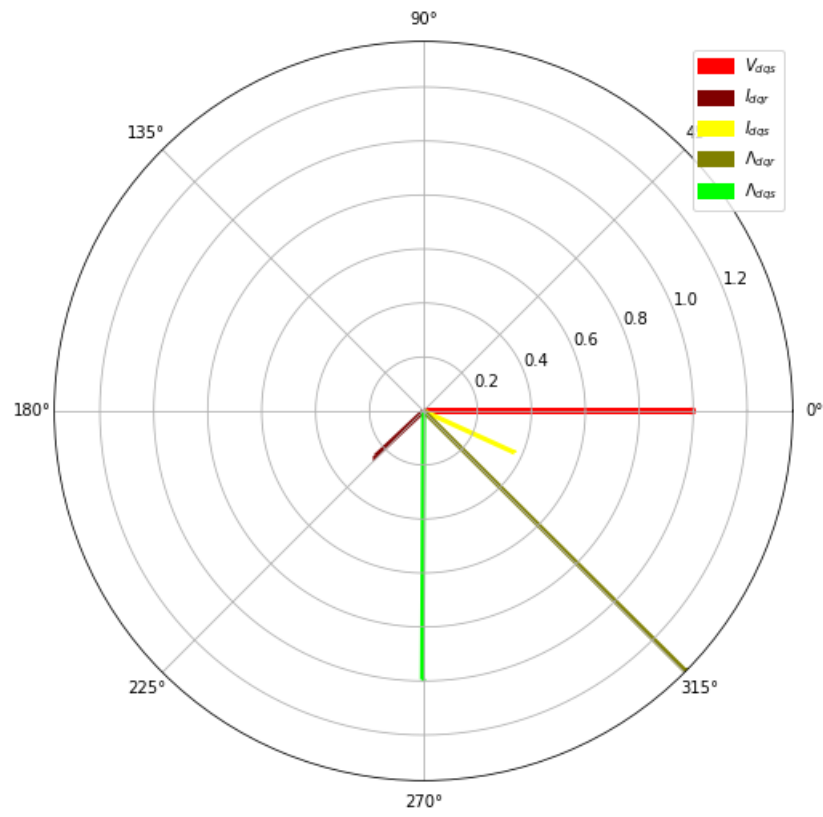
Sub-Part 1:

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

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

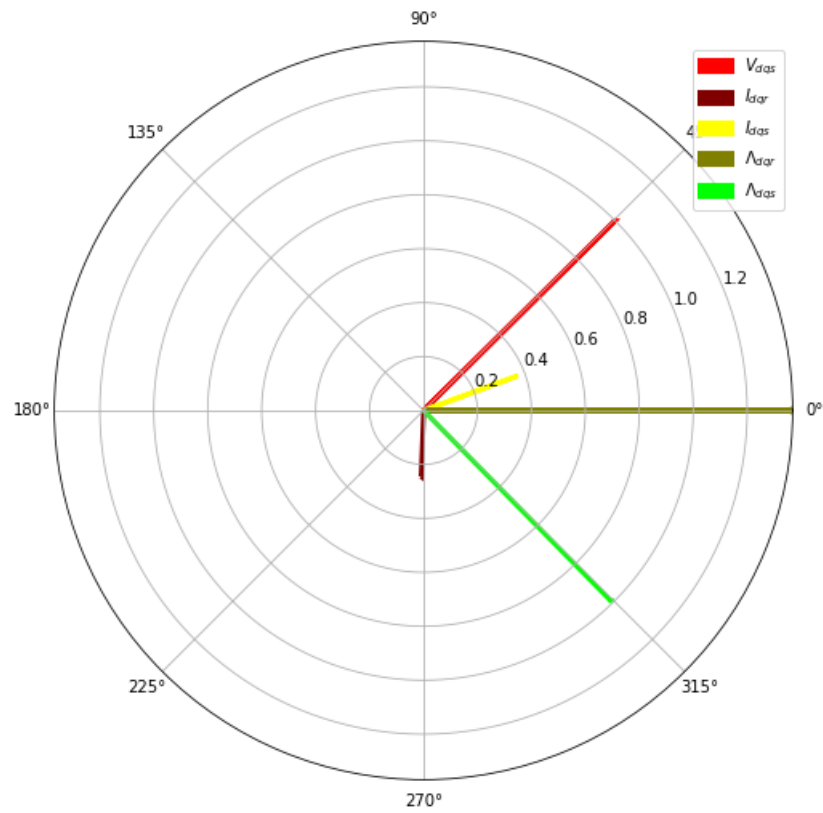
Sub-Part 2:

I.A.2 Phasor Plot



Sub-Part 3:

I.A.3 Phasor Plot



In [29]:

```
1  # Define Provided Machine Parameters
2  rs = 0.03 #pu
3  LLs = 0.1 #pu
4  Lm = 2.0 #pu
5  LLr = 0.1 #pu
6  rr = 0.03 #pu
7
8  # Define Rated Criteria
9  VdqsMag = 1
10 Tem = 1
11 wes = 1
12
13 # Assumptions
14 p = 4
15 at = 2
16
17 # Calculate Additional Inductance Terms
18 Ls = LLs + Lm
19 Lr = LLr + Lm
20 Lmr = 2/3 * (Lr - LLr)
21 Lsr = Lmr * at
22
23 # Define Equations Function as Solver
24 def equations(val):
25     Idr, Iqr, Ids, Iqs, LAMdr, LAMqr, LAMds, LAMqs, wr = val
26     A = (rs*Ids - wes*LAMqs) - VdqsMag
27     B = rs*Iqs - wes*LAMds
28     C = rr*Idr - (wes-wr)*LAMqr
29     D = rr*Iqr + (wes-wr)*LAMdr
30     E = (Ls*Ids + 3/2*Lsr*Idr) - LAMds
31     F = (Ls*Iqs + 3/2*Lsr*Iqr) - LAMqs
32     G = (3/2*Lsr*Ids+Lr*Idr) - LAMdr
33     H = (3/2*Lsr*Iqs+Lr*Iqr) - LAMqr
34     I = (3*p/4*(3/2*Lsr)/Lr*(LAMdr*Iqs-LAMqr*Ids)) - Tem
35     return(A,B,C,D,E,F,G,H,I)
36
37 #####
38 # A.1)
39
40 # Define Initial Guesses
41 Idr0 = -1
42 Iqr0 = -1
43 Ids0 = 1
44 Iqs0 = -1
45 LAMdr0 = 3/2*Lsr*Ids0 + Lr*Idr0
46 LAMqr0 = 3/2*Lsr*Iqs0 + Lr*Iqr0
47 LAMds0 = Ls*Ids0 + 3/2*Lsr*Idr0
48 LAMqs0 = Ls*Iqs0 + 3/2*Lsr*Iqr0
49 wr = 360
50
51 # Use Iterative Solver to Find Results
52 Idr, Iqr, Ids, Iqs, LAMdr, LAMqr, LAMds, LAMqs, wr = fsolve(equations, (
53     Idr0, Iqr0, Ids0, Iqs0, LAMdr0, LAMqr0, LAMds0, LAMqs0, wr))
54
55 # Define Complex Composition Function
56 def complexcomposer(Idr, Iqr, Ids, Iqs, LAMdr, LAMqr, LAMds, LAMqs, Vds, Vqs):
57     # Return order: Vdqs, Idqr, Idqs, LAMdqr, LAMdqs
58     return(np.array([
59         Vds + 1j*Vqs,
60         Idr + 1j*Iqr,
61         Ids + 1j*Iqs,
62         LAMdr+1j*LAMqr,
63         LAMds+1j*LAMqs,
64     ]))
65 Vds = VdqsMag
66 Vqs = 0
67 clist = complexcomposer(Idr, Iqr, Ids, Iqs, LAMdr, LAMqr, LAMds, LAMqs, Vds, Vqs)
68
69 # Generate into Latex Vector
70 vect_desc = (r"$$\begin{bmatrix}V_{\text{dqs}}\\I_{\text{dqr}}\\I_{\text{dqs}}\\L_{\text{dqr}}\\L_{\text{dqs}}\end{bmatrix}"+
71     r"I_{\text{dqs}}\\L_{\text{dqr}}\\L_{\text{dqs}}\end{bmatrix}"+
```

```

72         "\end{bmatrix}=")
73 P1_latex = vect_desc + ep.clatex(clist,predollar=False,double=True)
74
75 # Calculate Desired Terms
76 s_rated = round((wes-wr)/wes,5)
77 w_rated = round(wr,5)
78 lamdr_rated = round(abs(LAMdqr),5)
79
80 #####
81 # A.2)
82 # Use Complex Values to Plot Phasor Diagram
83 labels = [
84     "$V_{dqs}$",
85     "$I_{dqr}$",
86     "$I_{dqs}$",
87     "$\\Lambda_{dqr}$",
88     "$\\Lambda_{dqs}$",
89 ]
90 ep.phasorplot(clist,"I.A.2 Phasor Plot",labels,filename="I-A-2",size=8,linewidth=3,plot=debug)
91
92 #####
93 # A.3)
94 # Calculate Phase Shift
95 shift = np.angle(LAMdqr,deg=True)
96 # Shift Phasors by Phase Shift Specified
97 clist *= ep.phs(-shift)
98 # Use Complex Values to Plot Phasor Diagram
99 ep.phasorplot(clist,"I.A.3 Phasor Plot",labels,filename="I-A-3",size=8,linewidth=3,plot=debug)

```

Part B:

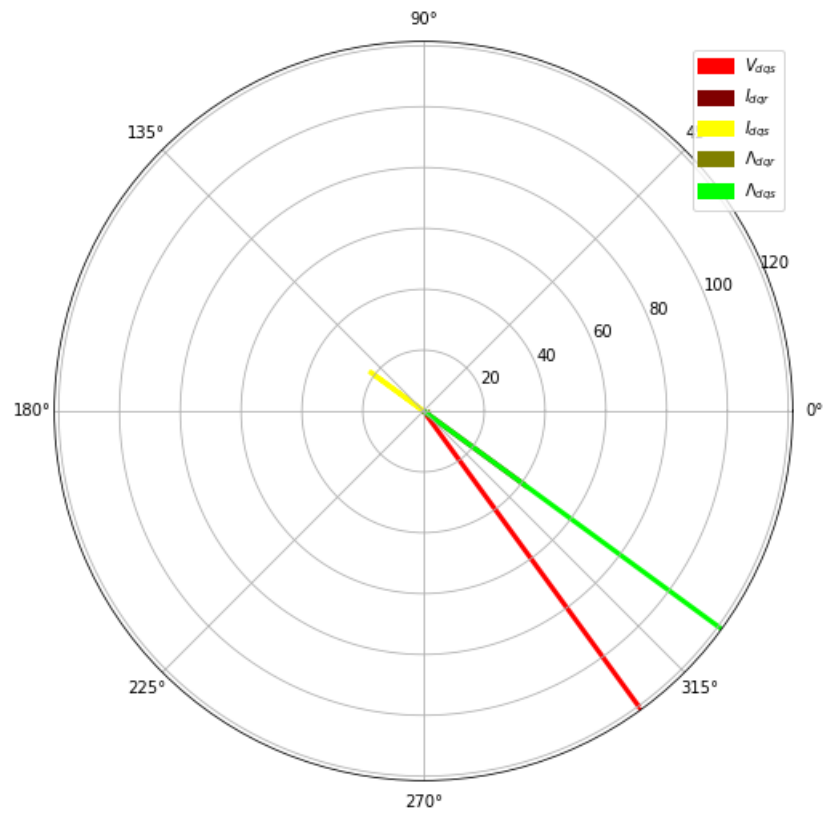
$$1) \quad \omega_{rB1} := 0.0 \quad \text{and} \quad \Lambda_{drB1} = \Lambda_{dr_rated}$$

$$2) \quad \omega_{rB2} = \omega_{r_rated} \quad \text{and} \quad \Lambda_{drB2} = \Lambda_{dr_rated}$$

$$3) \quad \omega_{rB3} = 2 \cdot \omega_{r_rated} \quad \text{and} \quad \Lambda_{drB3} = \frac{\Lambda_{dr_rated}}{2}$$

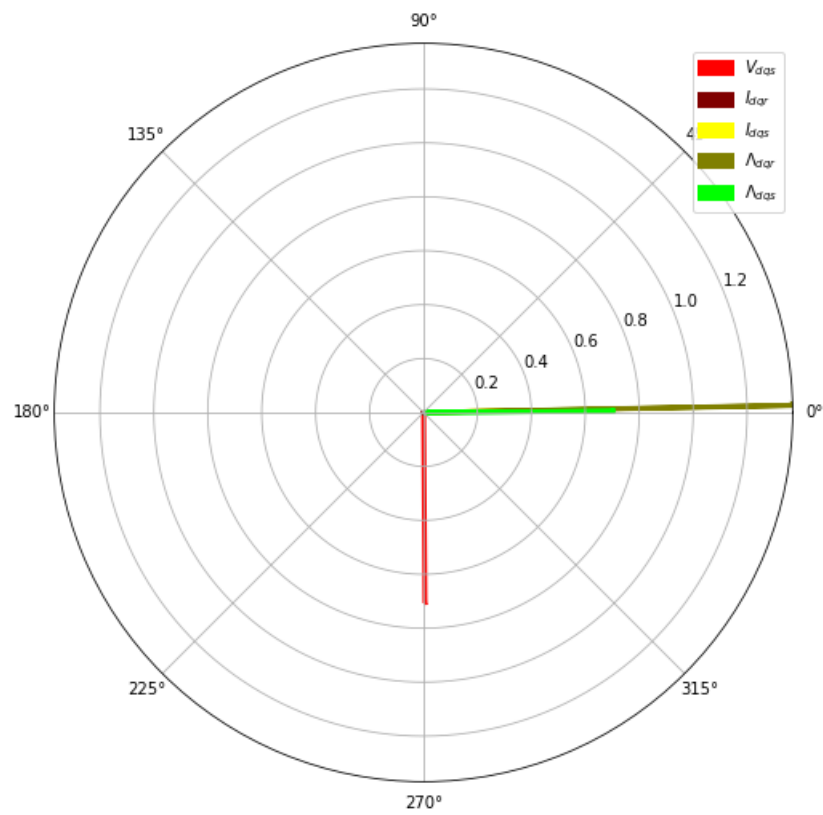
Sub-Part 1:

I.B.1 Phasor Plot

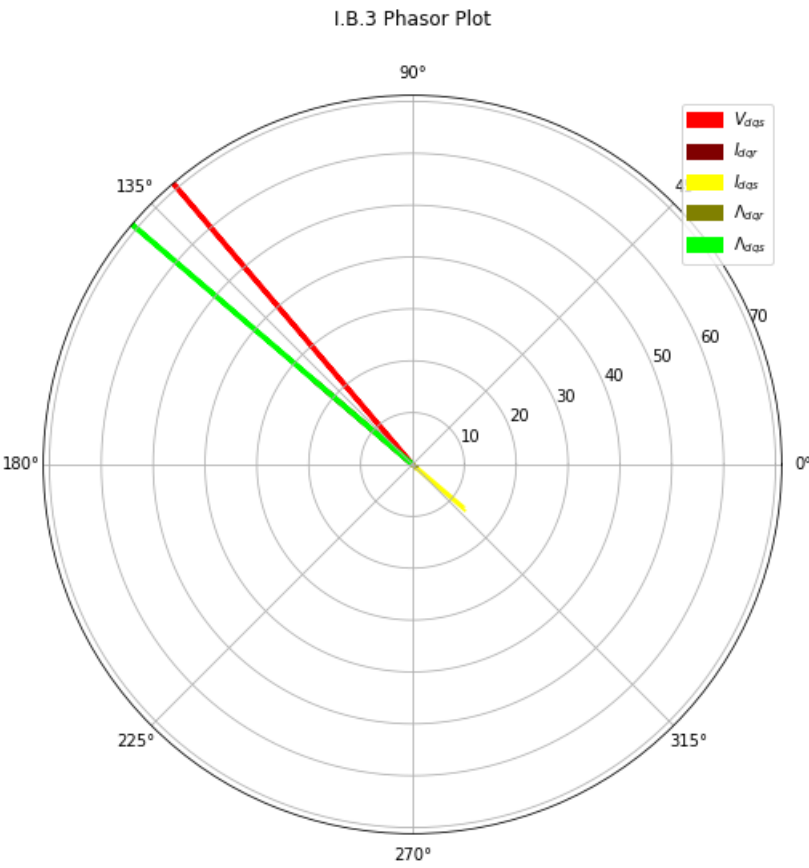


Sub-Part 2:

I.B.2 Phasor Plot



Sub-Part 3:



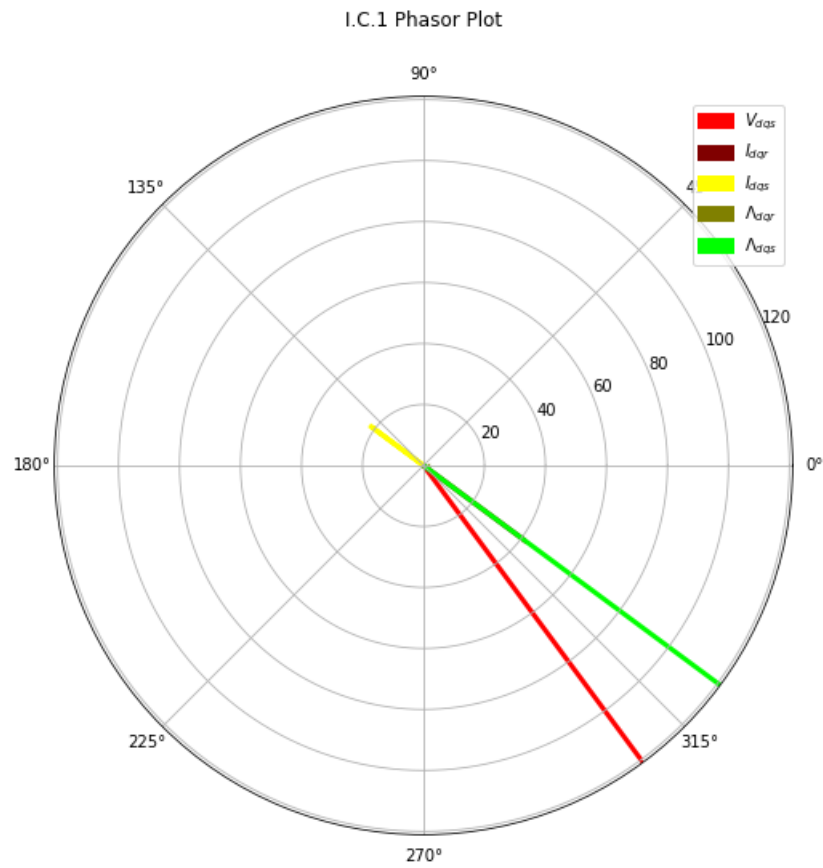
In [30]:

```
1  # Re-Define Known Values
2  Tem = 0
3
4  # Define Equations Function as Solver
5  def equations_B(val,wr,LAMdr):
6      Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = val
7      A = (rs*Ids - wes*LAMqs) - Vds
8      B = (rs*Iqs - wes*LAMds) - Vqs
9      C = rr*Idr - (wes-wr)*LAMqr
10     D = rr*Iqr + (wes-wr)*LAMdr
11     E = (Ls*Ids + 3/2*Lsr*Idr) - LAMds
12     F = (Ls*Iqs + 3/2*Lsr*Iqr) - LAMqs
13     G = (3/2*Lsr*Ids+Lr*Idr) - LAMdr
14     H = (3/2*Lsr*Iqs+Lr*Iqr) - LAMqr
15     I = (3*p/4*(3/2*Lsr)/Lr*(LAMdr*Iqs-LAMqr*Ids)) - Tem
16     return(A,B,C,D,E,F,G,H,I)
17
18 # Define Initial Guesses
19 Vds0 = 1
20 Vqs0 = 0
21 Idr0 = -1
22 Iqr0 = -1
23 Ids0 = 1
24 Iqs0 = -1
25 LAMqr0 = 3/2*Lsr*Iqs0 + Lr*Iqr0
26 LAMds0 = Ls*Ids0 + 3/2*Lsr*Idr0
27 LAMqs0 = Ls*Iqs0 + 3/2*Lsr*Iqr0
28
29 #####
30 # B.1)
31 wr = 0.0
32 LAMdr = lamdr_rated
33 b1 = lambda x: equations_B(x,wr,LAMdr)
34
35 # Use Iterative Solver to Find Results
36 Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(b1,(
37     Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))
38
39 # Generate Phasor Plot
40 clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
41 ep.phasorplot(clist,"I.B.1 Phasor Plot",labels,filename="I-B-1",size=8,linewidth=3,plot=debug)
42
43 #####
44 # B.2)
45 wr = w_rated
46 LAMdr = lamdr_rated
47 b2 = lambda x: equations_B(x,wr,LAMdr)
48
49 # Use Iterative Solver to Find Results
50 Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(b2,(
51     Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))
52
53 # Generate Phasor Plot
54 clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
55 ep.phasorplot(clist,"I.B.2 Phasor Plot",labels,filename="I-B-2",size=8,linewidth=3,plot=debug)
56
57 #####
58 # B.3)
59 wr = 2*w_rated
60 LAMdr = lamdr_rated/2
61 b3 = lambda x: equations_B(x,wr,LAMdr)
62
63 # Use Iterative Solver to Find Results
64 Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(b3,(
65     Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))
66
67 # Generate Phasor Plot
68 clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
69 ep.phasorplot(clist,"I.B.3 Phasor Plot",labels,filename="I-B-3",size=8,linewidth=3,plot=debug)
```


Part C:

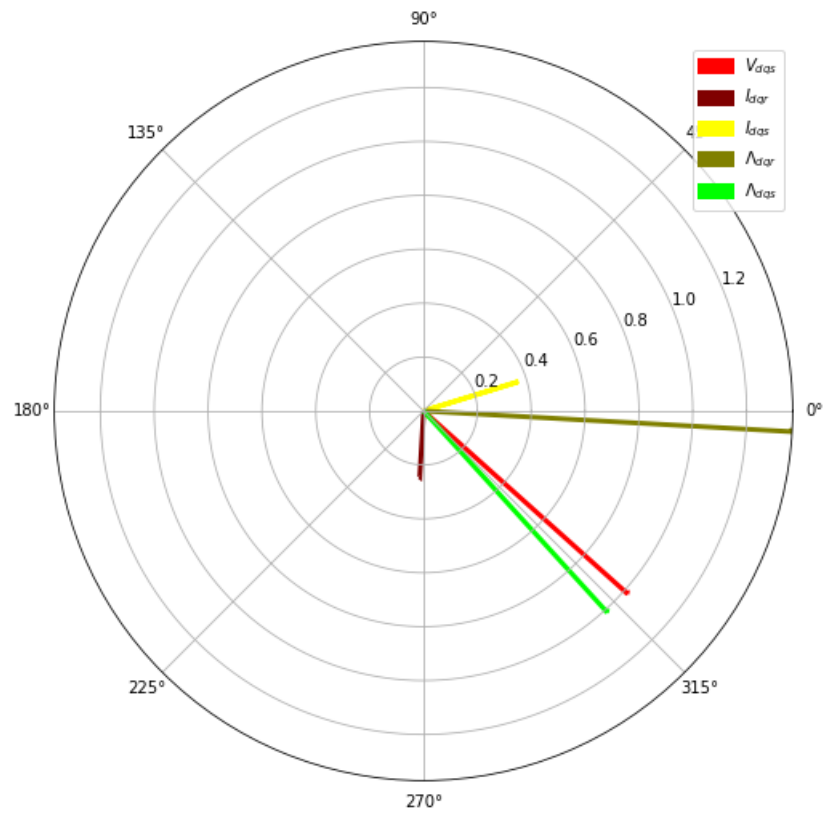
- 1) $T_{EMC1} := 1.0 \cdot \text{pu}$ $\omega_{rC1} := 0.0$ and $\Lambda_{drC1} = \Lambda_{dr_rated}$
- 2) $T_{EMC2} := 1.0 \cdot \text{pu}$ $\omega_{rC2} = \omega_{r_rated}$ and $\Lambda_{drC2} = \Lambda_{dr_rated}$
- 3) $T_{EMC3} := 0.5 \cdot \text{pu}$ $\omega_{rC3} = 2 \cdot \omega_{r_rated}$ and $\Lambda_{drC3} = \frac{\Lambda_{dr_rated}}{2}$

Sub-Part 1:



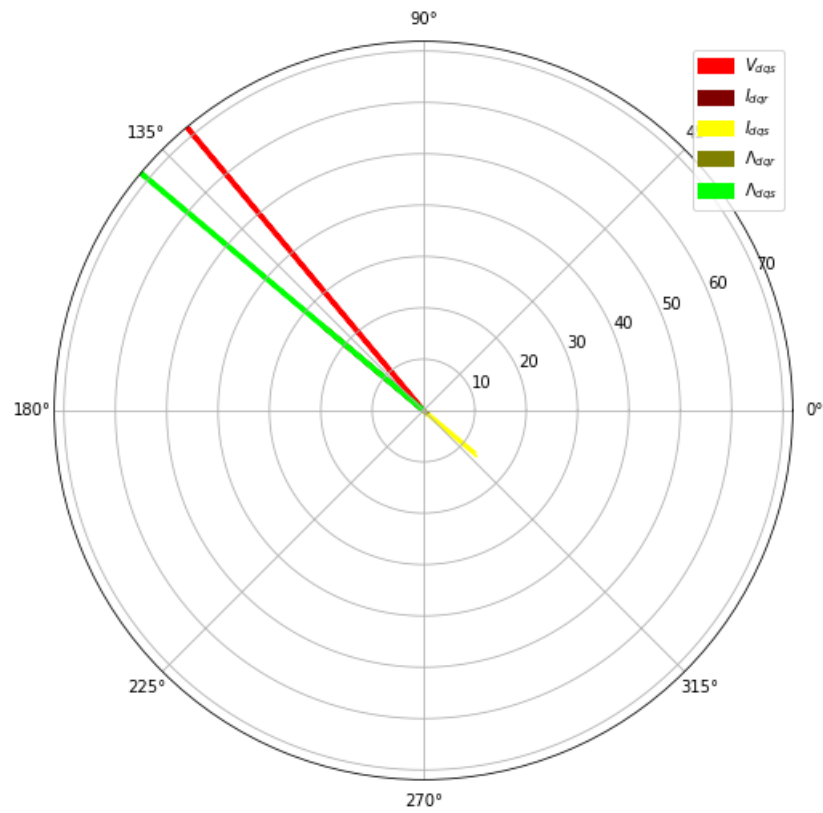
Sub-Part 2:

I.C.2 Phasor Plot



Sub-Part 3:

I.C.3 Phasor Plot



In [31]:

```
1  # Define Equations Function as Solver
2  def equations_C(val,wr,LAMdr,Tem):
3      Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = val
4      A = (rs*Ids - wes*LAMqs) - Vds
5      B = (rs*Iqs - wes*LAMds) - Vqs
6      C = rr*Idr - (wes-wr)*LAMqr
7      D = rr*Iqr + (wes-wr)*LAMdr
8      E = (Ls*Ids + 3/2*Lsr*Idr) - LAMds
9      F = (Ls*Iqs + 3/2*Lsr*Iqr) - LAMqs
10     G = (3/2*Lsr*Ids+Lr*Idr) - LAMdr
11     H = (3/2*Lsr*Iqs+Lr*Iqr) - LAMqr
12     I = (3*p/4*(3/2*Lsr)/Lr*(LAMdr*Iqs-LAMqr*Ids)) - Tem
13     return(A,B,C,D,E,F,G,H,I)
14
15  # Define Initial Guesses
16  Vds0 = 1
17  Vqs0 = 0
18  Idr0 = -1
19  Iqr0 = -1
20  Ids0 = 1
21  Iqs0 = -1
22  LAMqr0 = 3/2*Lsr*Iqs0 + Lr*Iqr0
23  LAMds0 = Ls*Ids0 + 3/2*Lsr*Idr0
24  LAMqs0 = Ls*Iqs0 + 3/2*Lsr*Iqr0
25
26  #####
27  # C.1)
28  Tem = 1.0
29  wr = 0.0
30  LAMdr = lamdr Rated
31  c = lambda x: equations_C(x,wr,LAMdr,Tem)
32
33  # Use Iterative Solver to Find Results
34  Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(c,(
35      Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))
36
37  # Generate Phasor Plot
38  clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
39  ep.phasorplot(clist,"I.C.1 Phasor Plot",labels,filename="I-C-1",size=8,linewidth=3,plot=debug)
40
41  #####
42  # C.2)
43  Tem = 1.0
44  wr = w Rated
45  LAMdr = lamdr Rated
46  c = lambda x: equations_C(x,wr,LAMdr,Tem)
47
48  # Use Iterative Solver to Find Results
49  Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(c,(
50      Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))
51
52  # Generate Phasor Plot
53  clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
54  ep.phasorplot(clist,"I.C.2 Phasor Plot",labels,filename="I-C-2",size=8,linewidth=3,plot=debug)
55
56  #####
57  # C.3)
58  Tem = 0.5
59  wr = 2*w Rated
60  LAMdr = lamdr Rated/2
61  c = lambda x: equations_C(x,wr,LAMdr,Tem)
62
63  # Use Iterative Solver to Find Results
64  Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(c,(
65      Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))
66
67  # Generate Phasor Plot
68  clist = complexcomposer(Idr,Iqr,Ids,Iqs,LAMdr,LAMqr,LAMds,LAMqs,Vds,Vqs)
69  ep.phasorplot(clist,"I.C.3 Phasor Plot",labels,filename="I-C-3",size=8,linewidth=3,plot=debug)
```

In []:

1	
---	--