Joe Stanley

ECE522 - EXAM1

In [2]:
```
# 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 III:

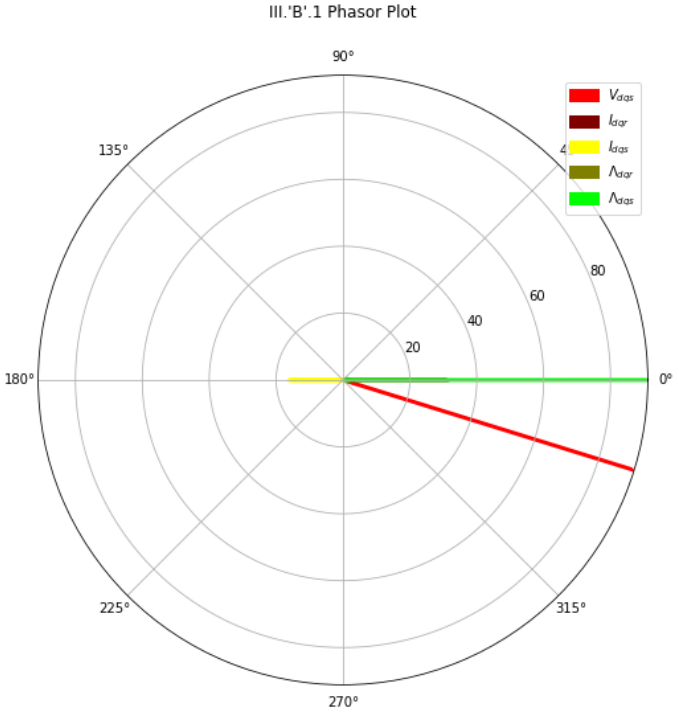
Repeat Parts B and C of Problem I for the situation where theparameter Lr/rr in the "slip calculator" is in error by -25%.

Comment on the effect on steady state performance of such "detuning" of the controller.

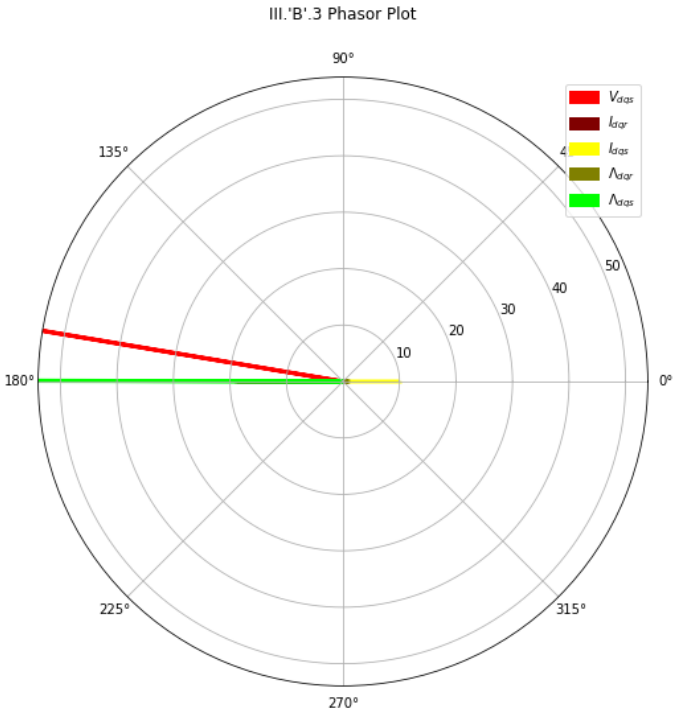
Just like Problem II, since we know: $s = \frac{\omega_{es} - \omega_r}{\omega_{es}}$, we can manipulate the equation into the form: $(\omega_{es} - \omega_r) = s \cdot \omega_{es}$. In this form, we can substitute it into our equations to solve.

Part 'B' (since we're only repeating parts B and C of problem 1):

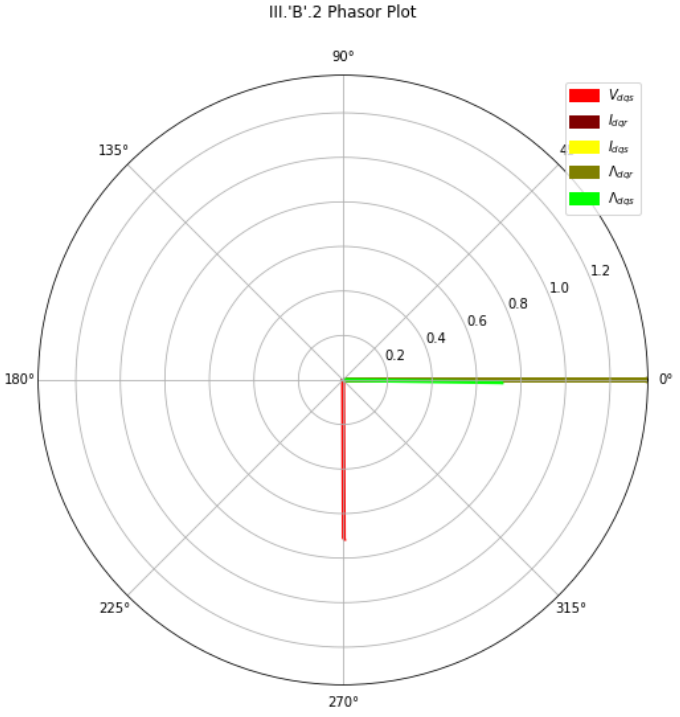
Sub-Part 1:



Sub-Part 2:


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

Sub-Part 3:


III.'B'.2 Phasor Plot
90°
135°
180°
0°
225°
315°
270°
0.2
0.4
0.6
0.8
1.0
1.2
V_{dqs}
I_{dqr}
I_{dqs}
Λ_{dqr}
Λ_{dqs}

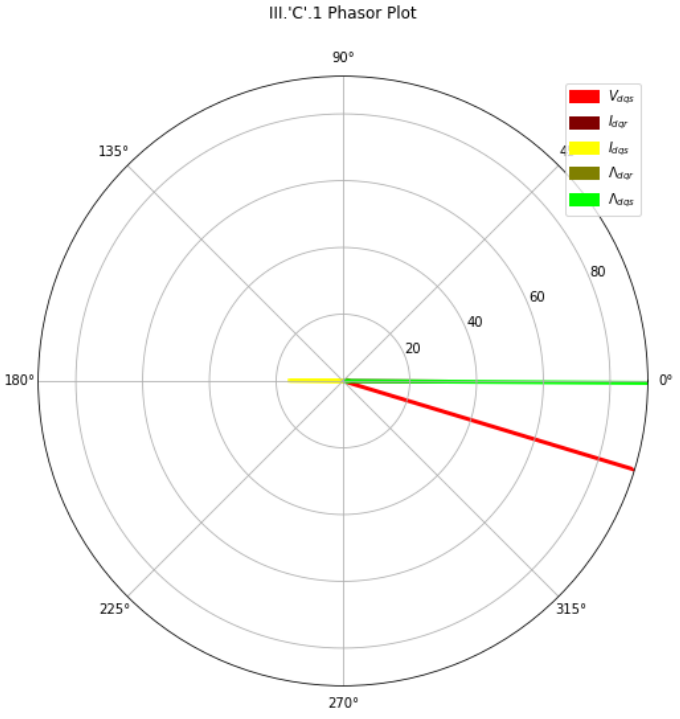
In [4]:

```
# 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
wes = 1
Tem = 0

# Assumptions
p = 4
at = 2

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

#####################################################
# Calculated from Problem 1
lamdr_rated = 1.36883
w_rated = 0.99468
#####################################################

# Define Equations Function as Solver
def equations_B(val,wr,LAMdr):
    Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = val
    ##############################
    s = (wes - wr) / wes # Find S
    s *= 0.75 # Apply -25% Error
    ##############################
    A = (rs*Ids - wes*LAMqs) - Vds
    B = (rs*Iqs - wes*LAMds) - Vqs
    C = rr*Idr - (wes*s)*LAMqr
    D = rr*Iqr + (wes*s)*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

# Define Plotting Labels
labels = [
    "$V_{dqs}$",
    "$I_{dqr}$",
    "$I_{dqs}$",
    "$\\Lambda_{dqr}$",
    "$\\Lambda_{dqs}$",
]

#########################################################################
# "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 = ep.compose([[Vds,Vqs],[Idr,Iqr],[Ids,Iqs],[LAMdr,LAMqr],[LAMds,LAMqs]])
clist *= ep.phs(-np.angle(clist[3],deg=True))
ep.phasorplot(clist,"III.'B'.1 Phasor Plot",labels,filename="III-B-1",size=8,linewidth=3,plot=debug)

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

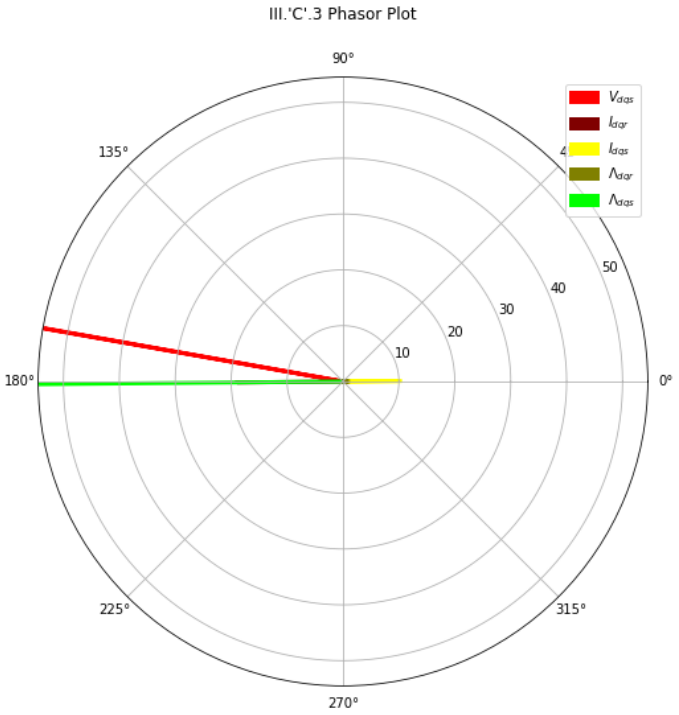
```
87
88  # Use Iterative Solver to Find Results
89  Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(b2,(
90      Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))
91
92  # Generate Phasor Plot
93  clist = ep.compose([[Vds,Vqs],[Idr,Iqr],[Ids,Iqs],[LAMdr,LAMqr],[LAMds,LAMqs]])
94  clist *= ep.phs(-np.angle(clist[3],deg=True))
95  ep.phasorplot(clist,"III.'B'.2 Phasor Plot",labels,filename="III-B-2",size=8,linewidth=3,plot=debug)
96
97  ########################################################################
98  # "B".3)
99  wr = 2*w_rated
100 LAMdr = lamdr_rated/2
101 b3 = lambda x: equations_B(x,wr,LAMdr)
102
103 # Use Iterative Solver to Find Results
104 Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = fsolve(b3,(
105     Idr0,Iqr0,Ids0,Iqs0,LAMqr0,LAMds0,LAMqs0,Vds0,Vqs0))
106
107 # Generate Phasor Plot
108 clist = ep.compose([[Vds,Vqs],[Idr,Iqr],[Ids,Iqs],[LAMdr,LAMqr],[LAMds,LAMqs]])
109 clist *= ep.phs(-np.angle(clist[3],deg=True))
110 ep.phasorplot(clist,"III.'B'.3 Phasor Plot",labels,filename="III-B-3",size=8,linewidth=3,plot=debug)
```

Part 'C' (since we're only repeating parts B and C of problem 1):

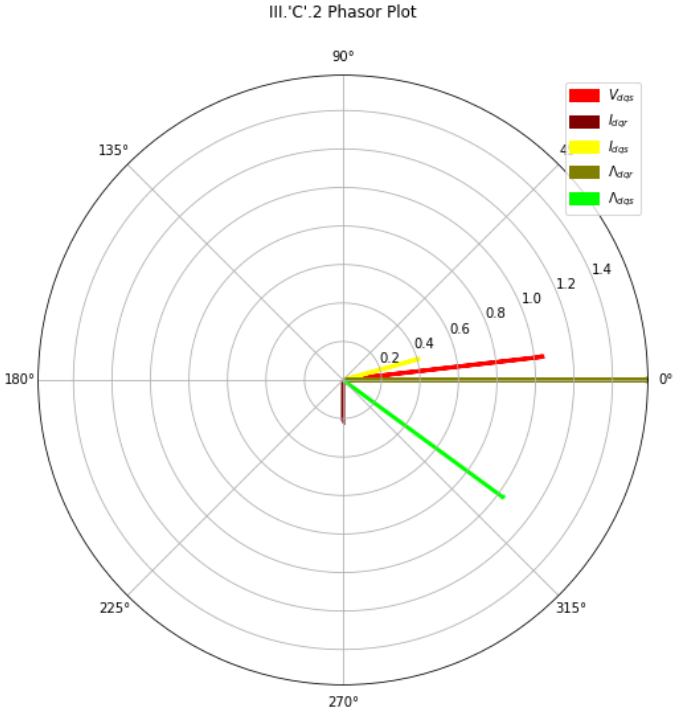
Sub-Part 1:



Sub-Part 2:



Sub-Part 3:



Comments and Analysis:

From comparison, it seems clear that these results are not too dissimilar from those found in the first problem (problem I). Perhaps the only truly notable difference is that magnitude difference between the results. Angle differences and general relations between the vectors appear to be largely the same between the Problem I results and these Problem III results. It is interesting to see that the magnitudes again seem to reflect the error in slip. It seems that perhaps it could be drawn that slip is directly proportional to these terms.

In [5]:

```
# Define Equations Function as Solver
def equations_C(val,wr,LAMdr,Tem):
    Idr,Iqr,Ids,Iqs,LAMqr,LAMds,LAMqs,Vds,Vqs = val
    ##############################
    s = (wes - wr) / wes # Find S
    s *= 0.75 # Apply -25% Error
    ##############################
    A = (rs*Ids - wes*LAMqs) - Vds
    B = (rs*Iqs - wes*LAMds) - Vqs
    C = rr*Idr - (wes*s)*LAMqr
    D = rr*Iqr + (wes*s)*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 = ep.compose([[Vds,Vqs],[Idr,Iqr],[Ids,Iqs],[LAMdr,LAMqr],[LAMds,LAMqs]])
clist *= ep.phs(-np.angle(clist[3],deg=True))
ep.phasorplot(clist,"III.'C'.1 Phasor Plot",labels,filename="III-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 = ep.compose([[Vds,Vqs],[Idr,Iqr],[Ids,Iqs],[LAMdr,LAMqr],[LAMds,LAMqs]])
clist *= ep.phs(-np.angle(clist[3],deg=True))
ep.phasorplot(clist,"III.'C'.2 Phasor Plot",labels,filename="III-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 = ep.compose([[Vds,Vqs],[Idr,Iqr],[Ids,Iqs],[LAMdr,LAMqr],[LAMds,LAMqs]])
clist *= ep.phs(-np.angle(clist[3],deg=True))
ep.phasorplot(clist,"III.'C'.3 Phasor Plot",labels,filename="III-C-3",size=8,linewidth=3,plot=debug)
```

In [ ]:

```
1
```