

Joe Stanley

ECE522 - HWK6

Problem:

Plot torque for time equal to 0 to 0.16 seconds for a three-phase short circuit on the machine of S22b handout. Use the complete not the approximate solution.

```
In [2]: 1 # Import Basic Libraries
        2 import numpy as np
        3 import matplotlib.pyplot as plt
        4 import electricpy as ep # Joe Stanley's Module, docs @: http://engineerjoe440.github.io/electricpy/
```

```
In [21]: 1 # Define Time Array
        2 time = np.linspace(0,0.16,1000)
        3
        4 # Define Machine Parameters
        5 rs = 0.018
        6 Lls = 0.1
        7 Llr = 0.1
        8 rr = 0.02
        9 rho = 4
       10 Lm = 2.0
       11
       12 # Define Calculated Machine Parameters
       13 Ls = Lls + Lm
       14 Lr = Llr + Lm
       15
       16 # Define ABC Base Parameters
       17 Sbase = ep.hp_to_watts(100)
       18 Vbase = 480
       19 Ibase = ep.ipu(Sbase,VLL=Vbase)
       20 Zbase = ep.zpu(Sbase,VLL=Vbase)
       21 webase = 2*np.pi*60
       22 Lbase = Zbase/webase
       23 wmbase = webase/(rho/2)
       24 tqbase = Sbase/wmbase
       25
       26 # Define QD Base Parameters
       27 Idqbase = np.sqrt(2)*Ibase
       28 Vdqbase = np.sqrt(2/3)*Vbase
       29
       30 # Define Machine Transient Parameters
       31 tq = Ls / (webase*rs)
       32 tqr = Lr / (webase*rr)
       33 alpha = tqr/tq
       34 phi = 1 - (Lm**2)/(Ls*Lr)
       35 tqsp = phi*tq
       36 tqrp = phi*tqr
       37
       38 # Define Eigenvalues
       39 wr = webase
       40 wrf = 0
```

In [51]:

```
1  # Define Induction Machine Eigenvalue Calculator
2  def imeigenvalues(Lr,Ls,Lm,Rr,Rs,wrf=0,freq=60):
3      """
4      Induction Machine Eigenvalue Calculator
5
6      Calculates the pertinent eigenvalues for an unloaded
7      induction machine given a specific set of machine
8      parameters.
9
10     Parameters
11     -----
12     Lr:      float
13             Inductance of the Rotor (in Henrys).
14     Ls:      float
15             Inductance of the Stator (in Henrys).
16     Lm:      float
17             Inductance of the Magnetizing branch
18             (in Henrys).
19     Rr:      float
20             Resistance of the Rotor (in Ohms).
21     Rs:      float
22             Resistance of the Stator (in Ohms).
23     wrf:     float, optional
24             Frequency (in radians/sec) of the rotor slip. **** Is this correct? I can't recall now... **
25             default=0
26     freq:    float, optional
27             Base frequency of the system (in Hertz).
28             default=60
29
30     Returns
31     -----
32     lam1:    complex
33             The First Eigenvalue
34     lam2:    complex
35             The Second Eigenvalue
36     """
37     # Calculate Required Values
38     omega_e_base = 2*np.pi*freq
39     omega_rf = wrf
40     torque_s = Ls/(omega_e_base*Rs)
41     torque_r = Lr/(omega_e_base*Rr)
42     alpha = torque_r / torque_s
43     phi = 1 - Lm**2/(Ls*Lr)
44     omega_r = omega_e_base
45     # Calculate k1
46     k1 = -1/(2*phi*torque_r)*(1+alpha)
47     k1 += 1j*(omega_r/2-omega_rf)
48     # Calculate k2
49     k2 = 1/(2*phi*torque_r)
50     k2 *= np.sqrt((1+alpha)**2-4*phi*alpha-(omega_r*phi*torque_r)**2
51                +2j*(alpha-1)*omega_r*phi*torque_r)
52     # Evaluate Eigenvalues and Return
53     lam1 = k1+k2
54     lam2 = k1-k2
55     return(lam1,lam2)
56
57 # Define IM 3-Phase SC Current Calculator
58 def imphs3sc(t,Is0,Lr,Ls,Lm,Rr,Rs,wrf=0,freq=60,real=True):
59     """
60     Induction Machine 3-Phase SC Calculator
61
62     Determines the short-circuit current at
63     a specified time for a three-phase fault
64     on an unloaded induction machine.
65
66     Parameters
67     -----
68     t:      array_like
69            The time at which to find the
70            current, may be int, float, or
71            numpy array.
72     Is0:    complex
73            The initial (t=0) current on
74            the stator.
75     Lr:     float
76            Inductance of the Rotor (in Henrys).
```

```

77     Ls:          float
78                Inductance of the Stator (in Henrys).
79     Lm:          float
80                Inductance of the Magnetizing branch
81                (in Henrys).
82     Rr:          float
83                Resistance of the Rotor (in Ohms).
84     Rs:          float
85                Resistance of the Stator (in Ohms).
86     wrf:         float, optional
87                Frequency (in radians/sec) of the rotor slip. **** Is this correct? I can't recall now... *
88                default=0
89     freq:        float, optional
90                Base frequency of the system (in Hertz).
91                default=60
92     real:        bool, optional
93                Control argument to force returned value
94                to be real part only. default=True
95
96     Returns
97     -----
98     ias:          array_like
99                Fault Current
100
101    """
102    # Calculate Required Values
103    omega_r = 2*np.pi*freq
104    torque_s = Ls/(omega_r*Rs)
105    phi = 1 - Lm**2/(Ls*Lr)
106    # Calculate Eigenvalues
107    lam1, lam2 = imeigenvalues(Lr,Ls,Lm,Rr,Rs,wrf,freq)
108    # Calculate pIs0
109    pIs0 = -(1/(phi*torque_s)+1j*(1-phi)/phi*omega_r)*Is0
110    # Calculate Constants
111    C1 = (lam2*Is0-pIs0)/(lam2-lam1)
112    C2 = (pIs0-lam1*Is0)/(lam2-lam1)
113    # Calculate ias and Return
114    ias = C1*np.exp(lam1*t)+C2*np.exp(lam2*t)
115    if real:
116        ias = np.real(ias)
117    return(ias)
118
119    # Define IM Torque Calculation
120    def impbs3sctorq(t,Is0,Lr,Ls,Lm,Rr,Rs,wrf=0,freq=60):
121        """
122        Parameters
123        -----
124        t:          array_like
125                The time at which to find the
126                current, may be int, float, or
127                numpy array.
128        Is0:         complex
129                The initial (t=0) current on
130                the stator.
131        Lr:          float
132                Inductance of the Rotor (in Henrys).
133        Ls:          float
134                Inductance of the Stator (in Henrys).
135        Lm:          float
136                Inductance of the Magnetizing branch
137                (in Henrys).
138        Rr:          float
139                Resistance of the Rotor (in Ohms).
140        Rs:          float
141                Resistance of the Stator (in Ohms).
142        p:           int
143                Number of electrical poles.
144        wrf:         float, optional
145                Frequency (in radians/sec) of the rotor slip. **** Is this correct? I can't recall now... *
146                default=0
147        freq:        float, optional
148                Base frequency of the system (in Hertz).
149                default=60
150
151        Returns
152        -----
153        Tem:         array_like

```

```

153         """ Induction machine torque in N*m
154
155     # Calculate Required Values
156     omega_r = 2*np.pi*freq
157     torque_s = Ls/(omega_r*Rs)
158     phi = 1 - Lm**2/(Ls*Lr)
159     # Calculate Eigenvalues
160     lam1, lam2 = imeigenvalues(Lr,Ls,Lm,Rr,Rs,wrf,freq)
161     # Calculate pIs0
162     pIs0 = -(1/(phi*torque_s)+1j*(1-phi)/phi*omega_r)*Is0
163     # Calculate Constants
164     C1 = (lam2*Is0-pIs0)/(lam2-lam1)
165     C2 = (pIs0-lam1*Is0)/(lam2-lam1)
166     # Calculate ias and Return
167     idqs = C1*np.exp(lam1*t)+C2*np.exp(lam2*t)
168     idqr = C2*np.exp(lam1*t)+C1*np.exp(lam2*t)
169     # Calculate Lambda
170     lamdqr = Lm*idqs+Lr*idqr
171     # Calculate Torque
172     Tem = Lm/Lr * (lamdqr.real*idqs.imag - lamdqr.imag*idqs.real)
173     return(Tem)

```

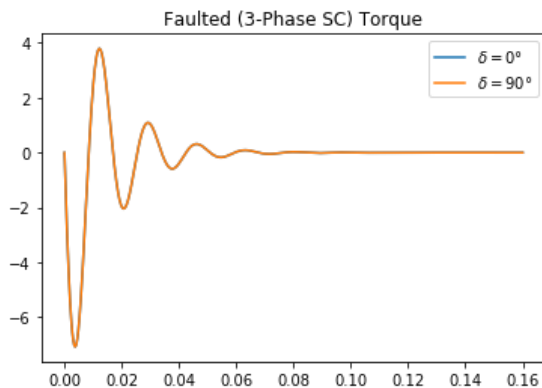
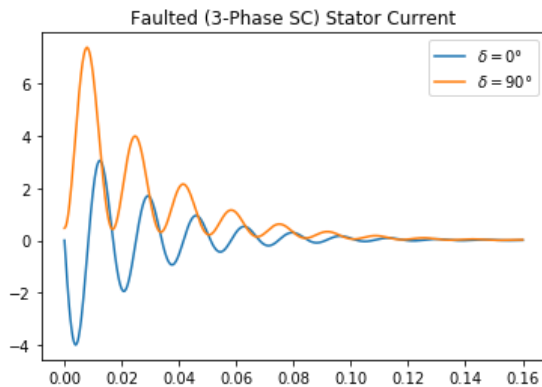
In [54]:

```

1  # Calculate Eigenvalues from New Function
2  lam1, lam2 = imeigenvalues(Lr,Ls,Lm,rr,rs) # Leave wrf and freq at defaults
3  print("\lambda1:",np.around(lam1,3),"\t\t\lambda2:",np.around(lam2,3))
4
5  # Define Source Voltage
6  Vs0 = ep.phasor(1,0)
7  Vs90 = ep.phasor(1,90)
8
9  # Calculate Stator Current at t=0 for both Cases
10 Is0 = Vs0/(rs+1j*Ls)
11 ep.cprint(Is0,label="Is (0°):")
12 Is90 = Vs90/(rs+1j*Ls)
13 ep.cprint(Is90,label="Is (90°):")
14
15 # Calculate ias over time
16 ias0 = imphs3sc(time,Is0,Lr,Ls,Lm,rr,rs)
17 ias90 = imphs3sc(time,Is90,Lr,Ls,Lm,rr,rs)
18
19 # Plot
20 plt.plot(time,ias0,label="\delta=0°")
21 plt.plot(time,ias90,label="\delta=90°")
22 plt.title("Faulted (3-Phase SC) Stator Current")
23 plt.legend()
24 plt.show()
25
26 # Calculate Torque over Time
27 Tq0 = imphs3sctorq(time,Is0,Lr,Ls,Lm,rr,rs)
28 Tq90 = imphs3sctorq(time,Is90,Lr,Ls,Lm,rr,rs)
29
30 # Plot
31 plt.plot(time,Tq0,label="\delta=0°")
32 plt.plot(time,Tq90,label="\delta=90°")
33 plt.title("Faulted (3-Phase SC) Torque")
34 plt.legend()
35 plt.show()

```

λ_1 : (-34.723+3.257j) λ_2 : (-38.653+373.734j)
 I_s (0°): 0.476 \angle -89.509°
 I_s (90°): 0.476 \angle 0.491°



In [38]:

1

In []:

1