Joe Stanley

ECE 525

In [12]:
```
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import eepower as eep
from eepower import u, m, k, M
```

In [84]:

```python
# Function Defenitions (to be used throughout assignment)

# Define CT Saturation Function
def ct_saturation(XR,Imag,Vrated,Irated,CTR,Rb,Xb,remnance=0,freq=60):
    # Define omega
    w = 2*np.pi*freq
    # Find Lb
    Lb = Xb/w
    # Re-evaluate Vrated
    Vrated = Vrated*(1-remnance)
    # Calculate each "term" (multiple)
    t1 = (1+XR)
    t2 = (Imag/(Irated*CTR))
    t3 = abs(Rb+1j*w*Lb)*100/Vrated
    # Evaluate
    result = t1*t2*t3
    # Test for saturation
    saturation = result >= 20
    # Return Results
    return(result,saturation)
# Define C-Class Calculator
def ct_cclass(XR,Imag,Irated,CTR,Rb,Xb,remnance=0,freq=60,satc=20):
    # Define omega
    w = 2*np.pi*freq
    # Find Lb
    Lb = Xb/w
    # Calculate each "term" (multiple)
    t1 = (1+XR)
    t2 = (Imag/(Irated*CTR))
    t3 = abs(Rb+1j*w*Lb)*100/satc
    # Evaluate
    Vr_w_rem = t1*t2*t3
    c_class = Vr_w_rem/(1-remnance)
    # Return Result
    return(c_class)
# Define Saturation Voltage at Rated Burden
def ct_satratburden(ALF,Inom,VArat=None,ANSIv=None):
    # Validate Inputs
    if VArat == None and ANSIv == None:
        raise ValueError("VArat or ANSIv must be specified.")
    elif VArat==None:
        # Calculate VArat from ANSIv
        Zrat = ANSIv/(20*Inom)
        VArat = Inom**2 * Zrat
    # Determine Vsaturation
    Vsat = ALF * VArat/Inom
    return(Vsat)
# Define CT Vpeak Formula
def ct_vpeak(Zb,Ip,N):
    return(np.sqrt(3.5*Zb*Ip*N))
# Define Saturation Time Calculator
def ct_timetosat(Vknee,XR,Rb,CTR,Imax,ts=None,npts=100,freq=60,plot=False):
    # Calculate omega
    w = 2*np.pi*freq
    # Calculate Tp
    Tp = XR/w
```

```python
    # If ts isn't specified, generate it
    if ts==None:
        ts = np.linspace(0,0.1,freq*npts)
    # Calculate inner term
    term = -XR*(np.exp(-ts/Tp)-1)
    # Calculate Vsaturation terms
    Vsat1 = Imax*Rb*(term+1)
    Vsat2 = Imax*Rb*(term-np.sin(w*ts))
    Vsat3 = Imax*Rb*(1-np.cos(w*ts))
    # If plotting requested
    if plot and isinstance(ts,np.ndarray):
        plt.plot(ts,Vsat1,label="Vsat1")
        plt.plot(ts,Vsat2,label="Vsat2")
        plt.plot(ts,Vsat3,label="Vsat3")
        plt.axhline(Vknee,label="V-knee",linestyle='--')
        plt.title("Saturation Curves")
        plt.xlabel("Time (ts)")
        plt.legend()
        plt.show()
    elif plot:
        print("Unable to plot a single point, *ts* must be a numpy-array.")
    # Determine the crossover points for each saturation curve
    Vsat1c = Vsat2c = Vsat3c = 0
    if isinstance(ts,np.ndarray):
        for i in range(len(ts)):
            if Vsat1.item(i)>Vknee and Vsat1c==0:
                Vsat1c = ts.item(i-1)
            if Vsat2.item(i)>Vknee and Vsat2c==0:
                Vsat2c = ts.item(i-1)
            if Vsat3.item(i)>Vknee and Vsat3c==0:
                Vsat3c = ts.item(i-1)
        results = (Vsat1c,Vsat2c,Vsat3c)
    else:
        results = (Vsat1,Vsat2,Vsat3)
    return(results)
```

Problem 1:

1. Recommend an appropriate CTR ratio for CT's connected at the terminals of a synchronous generator rated as follows: $P_{rated} = 500$ MW, $pf_{rated} = 0.8$, $V_{ratedLL} = 22$ kV, X/R = 40. The maximum available fault current is for a three phase fault, with a current level of 250 kA. Assume 5A secondary currents, and steps in primary side CT current ratings of 500 A for the range in question (so for example, ratios go up as 3000/5, 3500/5, on up).

In [85]:
```python
# Let's First Ignore the Max Current

# find I rated (primary)
x,x,S,x = eep.powerset(500*M,None,None,0.8)
print("Apparent Power (S):",S/M,"MVA")
Inom = S/(22*k)
print("Nominal Current:",Inom/k,"kA")
Iratprim = Inom*20
print("Rated Current (Primary):",Iratprim/k,"kA")

# We now see that the "rated current" is greater than max
CTR = np.ceil(Iratprim / 500) * 500
print("Resultant CTR:",str(CTR/k)+"k/5")
```

```
Apparent Power (S): 625.0 MVA
Nominal Current: 28.409090909090907 kA
Rated Current (Primary): 568.1818181818181 kA
Resultant CTR: 568.5k/5
```

Problem 2:

2. You need to determine the C-class rating for 2000/5 CTs applied on a transmission system. The X/R ratio for the fault impedance for the worst case fault seen by the CTs is 8. The CTs have a winding resistance of 0.0030 ohms/turn, relay is connected to the CT by 2500 feet of number 10 AWG, and the relay burden is 50mΩ. The magnitude of the impedance of the lead wire can be calculated using the formula below. A typical angle of the impedance of the lead wire is about 12-15 degrees (so highly resistive). For this problem you just need to use the magnitude.

- Recall that for 3 phase faults, you only need one length of lead wire, but for SLG and LL faults you need to include two lengths in lead resistance calculations

$$R_{AWG\_wire} = e^{0.232 \cdot Gauge - 2.32} \quad \text{in ohm/1000ft}$$

(a) Determine the C-class to never saturate if the decaying DC offset is neglected and the maximum fault current is 21000A (assume the worst case fault is SLG). Verify with the Mathcad CT model from class.

(b) Determine the C-class to never saturate with the decaying DC offset included and the maximum fault current is 21000A (assume you can go up to very high knee voltages). Again, verify with the Mathcad CT model from class.

(c) You are limited to using a C800 CT. Determine how long it will take the CT to saturate under the conditions of part (b). Again, verify with the Mathcad CT model.

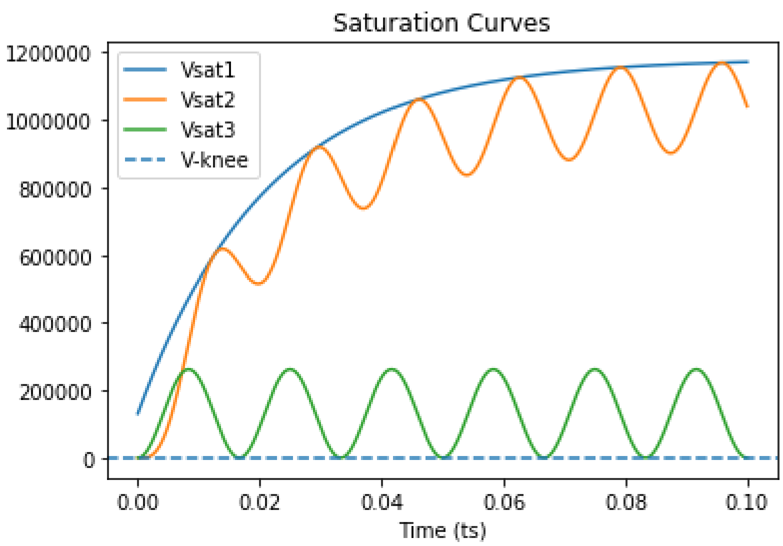
(d) Repeat parts (a)-(c) for a 3 phase fault of 27000 A with the same burden and X/R ratio.

In [86]:

```
# Define Givens:
CTR = 2000/5 #unitless
Irated = 5
N = CTR # Evaluated CTR
XRratio = 8 #unitless
R_CT = 0.003 #ohms per turn
RLY_dist = 2500 #ft
RLY_ga = 10 #gauge (AWG)
RLY_burden = 50*m #ohms
RLY_z_ang = range(12,16) # 12-15 inclusive of 15
Ifault = 21*k #A

# Define Wire Resistance Calculation:
def wireR(ga,ft):
    R1000 = np.exp(0.232*ga - 2.32)
    R = R1000*ft/1000
    return(R)

# Calculate Total Burden (Rb and Xb)
R_wire = wireR(RLY_ga,RLY_dist*2)
print("Wire Resistance:",R_wire,"Ω")
R_CT_full = R_CT*N
print("CT Resistance (Coils):",R_CT_full,"Ω")
Rb = R_wire + R_CT_full + RLY_burden
print("Resistive Burden:",Rb,"Ω")
Xb = Rb * 0.2 ################### From notes, use 20% of Rb for Xb
print("Inductive Burden:",Xb,"Ω")

# a) No DC offset
c = ct_cclass(0,Ifault,Irated,N,Rb,Xb)
print("Vrated:",c,"V")
print("a) C-Class Rating for NO DC-Offset:","C400")

# b) DC Offset
c = ct_cclass(XRratio,Ifault,Irated,N,Rb,Xb)
print("Vrated:",c,"V")
print("b) C-Class Rating for DC-Offset:","C"+str(int(round(c,-1))))

# c) Time to Saturate
Vs1, Vs2, Vs3 = ct_timetosat(800,XRratio,Rb,N,Ifault,plot=True)
print("Time to Saturate (Curve 1):",Vs1,"sec")
print("                 (Curve 2):",Vs2,"sec")
print("                 (Curve 3):",Vs3,"sec")
print("c) Time to Saturate:",Vs3/m,"mSec")

# Reset Fault Current for Second Part
Ifault = 27*k # A

# Re-evaluate Total Burden (for 3-Phase Fault)
R_wire = wireR(RLY_ga,RLY_dist)
print("Wire Resistance:",R_wire,"Ω")
R_CT_full = R_CT*N
print("CT Resistance (Coils):",R_CT_full,"Ω")
Rb = R_wire + R_CT_full + RLY_burden
print("Resistive Burden:",Rb,"Ω")
Xb = Rb * 0.2 ################### From notes, use 20% of Rb for Xb
```

```python
print("Inductive Burden:",Xb,"Ω")

c = ct_cclass(0,Ifault,Irated,N,Rb,Xb)
print("Vrated:",c,"V")
print("d) C-Class Rating for NO DC-Offset:","C200")
c = ct_cclass(XRratio,Ifault,Irated,N,Rb,Xb)
print("Vrated:",c,"V")
print("d) C-Class Rating for DC-Offset:","C"+str(int(round(c,-1))))
Vs1, Vs2, Vs3 = ct_timetosat(800,XRratio,Rb,N,Ifault)
print("Time to Saturate (Curve 1):",Vs1,"sec")
print("                  (Curve 2):",Vs2,"sec")
print("                  (Curve 3):",Vs3,"sec")
print("d) Time to Saturate:",Vs3/m,"mSec")
```

```
Wire Resistance: 5.0 Ω
CT Resistance (Coils): 1.2 Ω
Resistive Burden: 6.25 Ω
Inductive Burden: 1.25 Ω
Vrated: 334.62315558 V
a) C-Class Rating for NO DC-Offset: C400
Vrated: 3011.60840022 V
b) C-Class Rating for DC-Offset: C3010
```




```
Time to Saturate (Curve 1): 0.1 sec
                  (Curve 2): 0.0013502250375062512 sec
                  (Curve 3): 0.0002833805634272379 sec
c) Time to Saturate: 0.2833805634272379 mSec
Wire Resistance: 2.5 Ω
CT Resistance (Coils): 1.2 Ω
Resistive Burden: 3.75 Ω
Inductive Burden: 0.75 Ω
Vrated: 258.137862876 V
d) C-Class Rating for NO DC-Offset: C200
Vrated: 2323.24076588 V
d) C-Class Rating for DC-Offset: C2320
Time to Saturate (Curve 1): 0.1 sec
                  (Curve 2): 0.0014169028171361895 sec
                  (Curve 3): 0.0003333888981496916 sec
d) Time to Saturate: 0.3333888981496916 mSec
```

Problem 3:

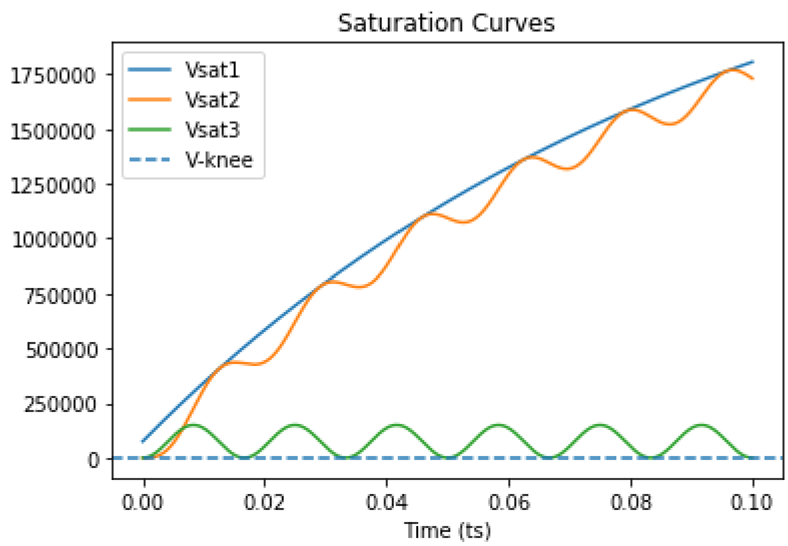
3. Repeat problem **2.** except with an X/R ratio of 35, and a fault current of 12000 A for a SLG fault and a 3 phase fault of 15000 A.

In [87]:

```python
# Define Givens:
CTR = 2000/5 #unitless
Irated = 5
N = CTR # Evaluated CTR
XRratio = 35 #unitless
R_CT = 0.003 #ohms per turn
RLY_dist = 2500 #ft
RLY_ga = 10 #gauge (AWG)
RLY_burden = 50*m #ohms
RLY_z_ang = range(12,16) # 12-15 inclusive of 15
Ifault = 12*k #A

# Define Wire Resistance Calculation:
def wireR(ga,ft):
    R1000 = np.exp(0.232*ga - 2.32)
    R = R1000*ft/1000
    return(R)

# Calculate Total Burden (Rb and Xb)
R_wire = wireR(RLY_ga,RLY_dist*2)
print("Wire Resistance:",R_wire,"Ω")
R_CT_full = R_CT*N
print("CT Resistance (Coils):",R_CT_full,"Ω")
Rb = R_wire + R_CT_full + RLY_burden
print("Resistive Burden:",Rb,"Ω")
Xb = Rb * 0.2 ################### From notes, use 20% of Rb for Xb
print("Inductive Burden:",Xb,"Ω")

# a) No DC offset
c = ct_cclass(0,Ifault,Irated,N,Rb,Xb)
print("Vrated:",c,"V")
print("a) C-Class Rating for NO DC-Offset:","C400")

# b) DC Offset
c = ct_cclass(XRratio,Ifault,Irated,N,Rb,Xb)
print("Vrated:",c,"V")
print("b) C-Class Rating for DC-Offset:","C"+str(int(round(c,-1))))

# c) Time to Saturate
Vs1, Vs2, Vs3 = ct_timetosat(800,XRratio,Rb,N,Ifault,plot=True)
print("Time to Saturate (Curve 1):",Vs1,"sec")
print("                 (Curve 2):",Vs2,"sec")
print("                 (Curve 3):",Vs3,"sec")
print("c) Time to Saturate:",Vs3/m,"mSec")

# Reset Fault Current
Ifault = 15*k # A

# Re-evaluate Total Burden (for 3-Phase Fault)
R_wire = wireR(RLY_ga,RLY_dist)
print("Wire Resistance:",R_wire,"Ω")
R_CT_full = R_CT*N
print("CT Resistance (Coils):",R_CT_full,"Ω")
Rb = R_wire + R_CT_full + RLY_burden
print("Resistive Burden:",Rb,"Ω")
Xb = Rb * 0.2 ################### From notes, use 20% of Rb for Xb
```

```python
print("Inductive Burden:",Xb,"Ω")

c = ct_cclass(0,Ifault,Irated,N,Rb,Xb)
print("Vrated:",c,"V")
print("d) C-Class Rating for NO DC-Offset:","C200")
c = ct_cclass(XRratio,Ifault,Irated,N,Rb,Xb)
print("Vrated:",c,"V")
print("d) C-Class Rating for DC-Offset:","C"+str(int(round(c,-1))))
Vs1, Vs2, Vs3 = ct_timetosat(800,XRratio,Rb,N,Ifault)
print("Time to Saturate (Curve 1):",Vs1,"sec")
print("                  (Curve 2):",Vs2,"sec")
print("                  (Curve 3):",Vs3,"sec")
print("d) Time to Saturate:",Vs3/m,"mSec")
```

```
Wire Resistance: 5.0 Ω
CT Resistance (Coils): 1.2 Ω
Resistive Burden: 6.25 Ω
Inductive Burden: 1.25 Ω
Vrated: 191.21323176 V
a) C-Class Rating for NO DC-Offset: C400
Vrated: 6883.67634335 V
b) C-Class Rating for DC-Offset: C6880
```




```
Time to Saturate (Curve 1): 0.1 sec
                 (Curve 2): 0.0011335222537089515 sec
                 (Curve 3): 0.00038339723287214534 sec
c) Time to Saturate: 0.38339723287214533 mSec
Wire Resistance: 2.5 Ω
CT Resistance (Coils): 1.2 Ω
Resistive Burden: 3.75 Ω
Inductive Burden: 0.75 Ω
Vrated: 143.40992382 V
d) C-Class Rating for NO DC-Offset: C200
Vrated: 5162.75725751 V
d) C-Class Rating for DC-Offset: C5160
Time to Saturate (Curve 1): 0.1 sec
                 (Curve 2): 0.0012502083680613436 sec
                 (Curve 3): 0.00043340556759459913 sec
d) Time to Saturate: 0.43340556759459914 mSec
```

Problem 4:

4. Suppose the CT with the characteristic described below is carrying 1128 A primary and is using a 1200/5 secondary tap and a resistive burden of 4 ohms. Neglecting the core loss resistance of the CT, (1) calculate the approximate initial voltage that would result across the CT secondaries if the CT secondary is accidentally opened (2) calculate the approximate final voltage it would reach if the insulation survives the initial overvoltage. Explain where the current is flowing.

ORIGIN := 1

CT Data: C600 class, 1200/5 Full ratio: $N_{full} := \frac{1200}{5}$ $N_{full} = 240$

CT Excitation Curve

$$\text{excitation} := \begin{pmatrix} .001 & 0.09 \\ .04 & 90 \\ .1 & 428 \\ .12 & 520 \\ .14 & 600 \\ .2 & 700 \\ .3 & 780 \\ .4 & 800 \\ 40 & 927 \end{pmatrix}$$

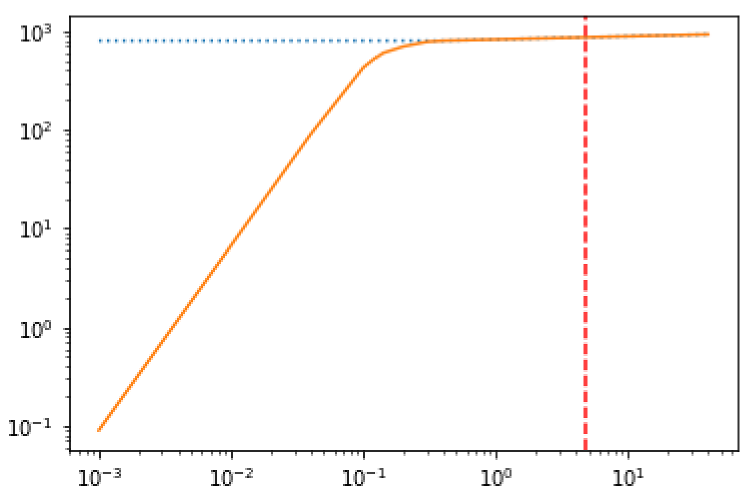
TAPS

$$t := \begin{pmatrix} 240 \\ 200 \\ 180 \\ 160 \\ 120 \\ 100 \\ 80 \\ 60 \\ 40 \\ 20 \end{pmatrix}$$

In [105]:

```
# Define givens
N = 1200/5
Ip = 1128
Rb = 4
c = 600
excitation = np.array([[0.001, 0.09],
                       [0.04,  90],
                       [0.1,   428],
                       [0.12,  520],
                       [0.14,  600],
                       [0.2,   700],
                       [0.3,   780],
                       [0.4,   800],
                       [40,    927]])

# Part 1:
# Finding the Peak Voltage, use pre-defined formula
Vmax = ct_vpeak(Rb,Ip,N)
print("1) Peak Voltage:",Vmax,"V")

# Define Line
def line(i):
    rise = excitation[8,1]-excitation[7,1]
    run = excitation[8,0]-excitation[7,0]
    return(rise/run*(i-excitation[7,0])+excitation[7,1])

# Plot V-vs-I
plt.plot(excitation[:,[0]],line(excitation[:,[0]]),linestyle=':')
plt.plot(excitation[:,[0]],excitation[:,[1]])
plt.xscale("log")
plt.yscale("log")
plt.axvline(Ip/N,linestyle='--',color='r')
plt.show()

# Evaluate Voltage at Specific Secondary Current
Isec = Ip / N
V2 = line(Isec) # Find Voltage
print("2) Final Voltage:",V2,"V")
```

```
1) Peak Voltage: 1946.8127799 V
```



2) Final Voltage: 813.79040404 V

TESTING...

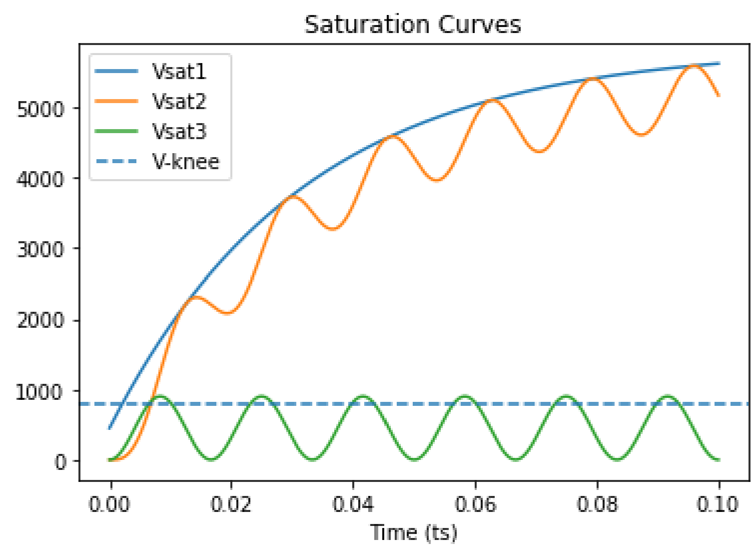
In [89]:

```
# Test From Session 6 Handout
Imag = 16000
X = 38
R = 2
Rb = 8
Xb = 1
Irat = 5
N = 240
Vrat = 800
# Print
print("Float:\t\tSaturated:")
print(ct_saturation(X/R,Imag,Vrat,Irat,N,Rb,Xb))
```

```
Float:          Saturated:
(268.74192494328497, True)
```

In [90]:

```
# Test From Session 8 Handout
x = (ct_timetosat(800,12,5,1200/5,90,plot=True))
print("Vs1\t\t\tVs2\t\t\tVs3")
print(x)
```




```
Vs1                     Vs2                     Vs3
(0.002117019503250542, 0.0068178029671611936, 0.006517752958826471)
```

In [92]:

```
x = np.array([1,2,3])
print(x[0])
x[1] = 5
print(x[1])
```

```
1
5
```

In [ ]:

```

```