Joe Stanley

ECE 525 - FINAL EXAM

In [1]:
```
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import eepower as eep
from eepower import p,n,u,m,k,M
```

In [2]:

```python
# Define CTR Printing Method
CT = lambda x: print(x[2],int(np.ceil(x[0])),"/",x[1]," ->\t",int(np.ceil(x[0]/x[1])))
# Define Transformer Shift Correction Matricies
XFMY0 = np.array([[1,0,0],[0,1,0],[0,0,1]])
XFMD1 = 1/np.sqrt(3) * np.array([[1,-1,0],[0,1,-1],[-1,0,1]])
XFMD11 = 1/np.sqrt(3) * np.array([[1,0,-1],[-1,1,0],[0,-1,1]])
XFM12 = 1/3 * np.array([[2,-1,-1],[-1,2,-1],[-1,-1,2]])
# Define TAP Calculator
def protectiontap(CTR,S,VLN=None,VLL=None):
    """
    protectiontap Function

    Evaluates the required TAP setting based on the rated power of
    a transformer (the object being protected) and the voltage
    (either primary or secondary) in conjunction with the CTR
    (current transformer ratio) for the side in question (primary/
    secondary).

    Parameters
    ----------
    CTR:        float
                The Current Transformer Ratio.
    S:          float
                Rated apparent power magnitude (VA/VAR/W).
    VLN:        float, exclusive
                Line-to-Neutral voltage in volts.
    VLL:        float, exclusive
                Line-to-Line voltage in volts.

    Returns
    -------
    TAP:        float
                The TAP setting required to meet the specifications.
    """
    # Condition Voltage(s)
    if VLL != None:
        V = abs(np.sqrt(3)*VLL)
    elif VLN != None:
        V = abs(3 * VLN)
    else:
        raise ValueError("One or more voltages must be provided.")
    # Calculate TAP
    TAP = abs(S) / (V*CTR)
    return(TAP)
# Define Current Correction Calculator
def correctedcurrents(Ipri,TAP,correction="Y",CTR=1):
    """
    correctedcurrents Function:

    Function to evaluate the currents as corrected for microprocessor-
    based relay protection schemes.

    Parameters
    ----------
    Ipri:       list of complex
                Three-phase set (IA, IB, IC) of primary currents.
    TAP:        float
                Relay's TAP setting.
    correction: string, optional
                String defining correction factor, may be one of:
                (Y, D+, D-, Z); Y denotes Y (Y0) connection, D+
                denotes Dab (D1) connection, D- denotes Dac (D11)
                connection, and Z (Z12) denotes zero-sequence
                removal. default="Y"
    CTR:        float
                Current Transformer Ratio, default=1

    Returns
    -------
    Isec_corr:  list of complex
                The corrected currents to perform operate/restraint
                calculations with.
    """
    # Define Matrix Lookup
    MAT = {   "Y"  : XFMY0,
              "D+" : XFMD1,
              "D-" : XFMD11,
              "Z"  : XFM12}
    # Condition Inputs
    Ipri = np.asarray(Ipri)
    if isinstance(correction,list):
```

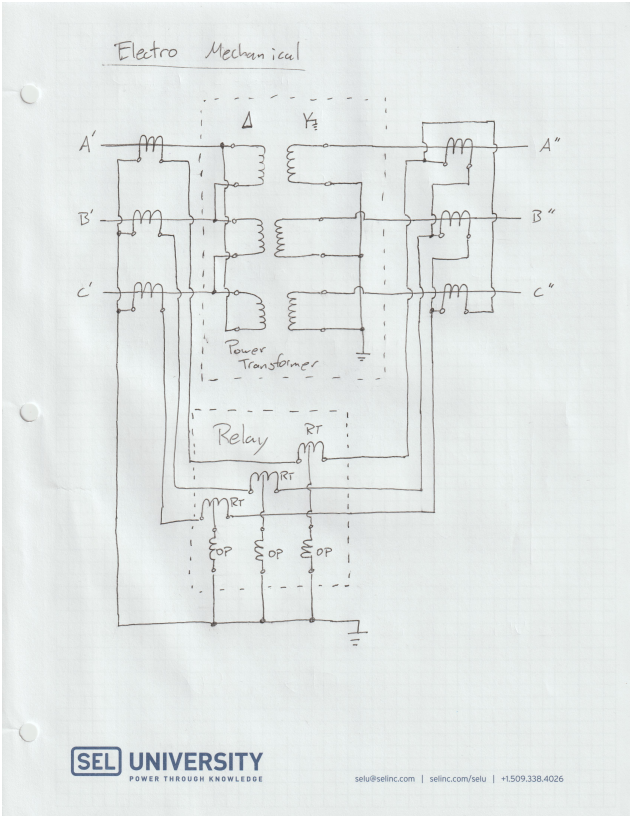
```
        mult = MAT[correction[0]]
        for i in correction[1:]:
            mult = mult.dot(MAT[i])
    elif isinstance(correction,str):
        mult = MAT[correction]
    elif isinstance(correction,np.ndarray):
        mult = correction
    else:
        raise ValueError("Correction must be string or list of strings.")
    # Evaluate Corrected Current
    Isec_corr = 1/TAP * mult.dot(Ipri/CTR)
    return(Isec_corr)
# Define Iop/Irt Calculator
def iopirt(IpriHV,IpriLV,TAPHV,TAPLV,corrHV="Y",corrLV="Y",CTRHV=1,CTRLV=1):
    """
    iopirt Function:

    Calculates the operating current (Iop) and the restraint
    current (Irt) as well as the slope.

    Parameters
    ----------
    IpriHV:     list of complex
                Three-phase set (IA, IB, IC) of primary currents
                on the high-voltage side of power transformer.
    IpriLV      list of complex
                Three-phase set (IA, IB, IC) of primary currents
                on the low-voltage side of power transformer.
    TAPHV       float
                Relay's TAP setting for high-voltage side of
                power transformer.
    TAPLV       float
                Relay's TAP setting for low-voltage side of
                power transformer.
    corrHV      string, optional
                String defining correction factor on high-voltage
                side of power transformer, may be one of:
                (Y, D+, D-, Z); Y denotes Y (Y0) connection, D+
                denotes Dab (D1) connection, D- denotes Dac (D11)
                connection, and Z (Z12) denotes zero-sequence
                removal. default="Y"
    corrLV      string, optional
                String defining correction factor on low-voltage
                side of power transformer, may be one of:
                (Y, D+, D-, Z); Y denotes Y (Y0) connection, D+
                denotes Dab (D1) connection, D- denotes Dac (D11)
                connection, and Z (Z12) denotes zero-sequence
                removal. default="Y"
    CTRHV       float
                Current Transformer Ratio for high-voltage side
                of power transformer, default=1
    CTRLV       float
                Current Transformer Ratio for low-voltage side
                of power transformer, default=1

    Returns
    -------
    Iop:        list of float
                The operating currents for phases A, B, and C.
    Irt:        list of float
                The restraint currents for phases A, B, and C.
    slope:      list of float
                The calculated slopes for phases A, B, and C.
    """
    # Calculate Corrected Currents
    IcorHV = correctedcurrents(IpriHV,TAPHV,corrHV,CTRHV)
    IcorLV = correctedcurrents(IpriLV,TAPLV,corrLV,CTRLV)
    # Calculate Operate/Restraint Currents
    Iop = np.absolute( IcorHV + IcorLV )
    Irt = np.absolute(IcorHV) + np.absolute(IcorLV)
    # Calculate Slopes
    slope = Iop/Irt
    return(Iop,Irt,slope)
```

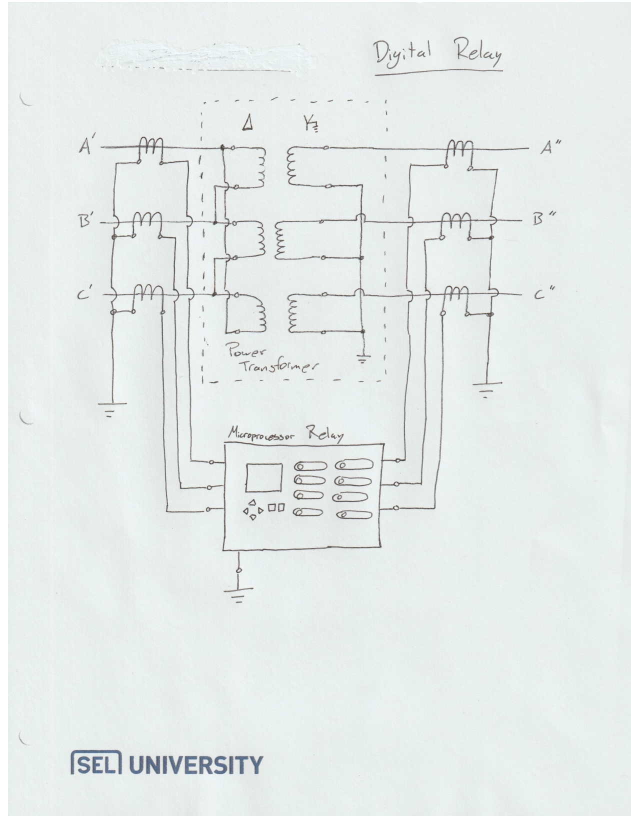
Problem 1:

A) Connection Diagrams:

Electro-Mechanical	Microprocessor

Electro-Mechanical	Microprocessor





In [3]:

```
# Define Parameters:
SrcXs1 = 0.4
SrcXoR = 30
SrcRg = 5/100
XfmX = 12/100
XfmXoR = 10
XfmN = eep.phaseline(Iline=24/345,complex=True)
ZL1 = eep.phasor(0.01,85.4)

# A)

# Evaluate Impedances
Zsrc1 = (SrcRg + SrcXs1 / SrcXoR + 1j*SrcXs1)*eep.zpu(S=250*M,VLL=24*k)
print("Source Impedance:",Zsrc1,"Ω")
Zxfm1 = (XfmX / XfmXoR + 1j*XfmX)*eep.zpu(S=250*M,VLL=24*k)
print("Transformer Impedance:",Zxfm1,"Ω")

# Evaluate Worst-Case Currents:
ILV = eep.phaseline(VLL=24*k)/Zsrc1
eep.cprint(ILV/k,"kA","Worst Case Current (Low-Side):")
IHV = eep.phaseline(VLL=24*k)/(Zsrc1+Zxfm1) * XfmN
eep.cprint(IHV/k,"kA","Worst Case Current (High-Side):")

# Calculate CT Ratios:
print("\nA)")
CT((abs(ILV),5,"Low-Side CTR:"))
CT((abs(IHV),5,"High-Side CTR:"))


# B)

# Calculate TAP settings
LvTAP = protectiontap(abs(ILV)/5,250*M,VLL=24*k)
HvTAP = protectiontap(abs(IHV)/5,250*M,VLL=345*k)
print("\nB)")
print("Low-Side TAP Setting:",LvTAP,"A")
print("High-Side TAP Setting:",HvTAP,"A")

# Demonstrate Correction Matricies
print("LV-Correction Matrix:\n",XFMD11,"\t (D11)")
print("HV-Correction Matrix:\n",XFMY0,"\t (Y0)")
```

```
Source Impedance: (0.14592+0.9216j) Ω
Transformer Impedance: (0.027648+0.27647999999999995j) Ω
Worst Case Current (Low-Side): 14.85 ∠ -81.003° kA
Worst Case Current (High-Side): 0.46 ∠ -51.757° kA

A)
Low-Side CTR: 14851 / 5  ->       2971
High-Side CTR: 460 / 5  ->       92

B)
Low-Side TAP Setting: 2.0249142643030043 A
High-Side TAP Setting: 4.550344309316969 A
LV-Correction Matrix:
 [[ 0.57735027  0.         -0.57735027]
 [-0.57735027  0.57735027  0.        ]
 [ 0.         -0.57735027  0.57735027]]          (D11)
HV-Correction Matrix:
 [[1 0 0]
 [0 1 0]
 [0 0 1]]        (Y0)
```

C) Pickup setting for the unrestrained overcurrent should be based on the fault current available for faults at 80% of the line (or less) which equates to what is commonly known as zone 1. To calculate these currents in a simple manner, we will use the three-phase fault current available. It's worth noting that this current should only be applied for the phase-protection. Ground and negative sequence currents should be calculated for the other protection elements (50G and 50Q).

In [4]:

```
# Evaluate the phase fault current
Zeq = Zsrc1+Zxfm1 + 0.8*ZL1*eep.zpu(S=250*M,VLL=24*k)
Ifaultzone1 = eep.phaseline(VLL=24*k)/Zeq
print("Fault Current:",abs(Ifaultzone1),"A")
print("Low-Side Pickup Setting:",abs(Ifaultzone1)/(abs(ILV)/5),"A-secondary")
print("High-Side Pickup Setting:",abs(Ifaultzone1 * XfmN)/(abs(IHV)/5),"A-secondary")
```

```
Fault Current: 11274.69721925425 A
Low-Side Pickup Setting: 3.7961502030179632 A-secondary
High-Side Pickup Setting: 4.925160415766922 A-secondary
```

D) Harmonic restraint and harmonic blocking are two methods primarily used to avoid erraneously tripping durring transformer energization (and subsequent inrush). Harmonic restraint was typically applied in electro-mechanical relays as a mode of restraining the operate coil and trip mechanism. Harmonic blocking is a method typically applied in microprocessor relays which monitors the level of second harmonic to make decisions on whether the transformer is currently being energized. These methods should be acting when the transformer is being energized and presents a significant load as itself. During these circumstances, the transformer differential protection will appear to have detected an internal fault. Although it is possible for an internal fault to actually be present, it is likely just the fact that most current is being consumed by the magnetization branch of the transformer to induce the flux appropriately.

E) Slope Setting Calculation

In [5]:

```
# Test with some arbitrary balanced current
Imag = 100
ILV_abc = np.array([eep.phasor(Imag,0),eep.phasor(Imag,-120),eep.phasor(Imag,-240)])
IHV_abc = -ILV_abc * XfmN * 0.9 # -10% of nominal turns ratio
eep.cprint(ILV_abc,label="Low-Side:")
eep.cprint(IHV_abc,label="High-Side:")

x,x,slp = iopirt(IHV_abc,ILV_abc,HvTAP,LvTAP,corrHV="D-",corrLV="Y",CTRHV=abs(IHV)/5,CTRLV=abs(ILV)/5)

print("Calculated Slopes:",slp*100,"%")
print("Selected Slope Setting:",np.ceil(slp[0]*100),"%")
```

```
[['Low-Side: 100.0 ∠ 0.0° ']
 ['Low-Side: 100.0 ∠ -120.0° ']
 ['Low-Side: 100.0 ∠ 120.0° ']]
[['High-Side: 3.615 ∠ -150.0° ']
 ['High-Side: 3.615 ∠ 90.0° ']
 ['High-Side: 3.615 ∠ -30.0° ']]
Calculated Slopes: [31.61226239 31.61226239 31.61226239] %
Selected Slope Setting: 32.0 %
```

In [6]:

```
# F)

# Define Case Currents
CASES = [
{"IHV" : np.array([eep.phasor(418.37,-175.84),eep.phasor(418.37,64.16),eep.phasor(418.37,-55.84)]),
"ILV" : np.array([eep.phasor(6014.06,-25.84),eep.phasor(6014.06,-145.84),eep.phasor(6014.06,94.158)])},
{"IHV" : np.array([eep.phasor(469.504,-175.84),eep.phasor(469.504,64.16),eep.phasor(469.504,-55.84)]),
"ILV" : np.array([eep.phasor(6224.557,-25.84),eep.phasor(6224.557,-145.84),eep.phasor(6224.557,94.158)])},
{"IHV" : np.array([eep.phasor(134.270,-129.79),eep.phasor(1*p,100.62),eep.phasor(136.266,-43.052)]),
"ILV" : np.array([eep.phasor(1580.052,3.132),eep.phasor(1141.957,-129.79),eep.phasor(1158.921,136.948)])},
{"IHV" : np.array([eep.phasor(1296.862,94.049),eep.phasor(341.511,97.217),eep.phasor(341.511,97.217)]),
"ILV" : np.array([eep.phasor(8106.92,-87.074),eep.phasor(8106.92,92.926),eep.phasor(1*p,104.036)])},
{"IHV" : np.array([eep.phasor(1296.862*0.2,94.049),eep.phasor(341.511,97.217),eep.phasor(341.511,97.217)]),
"ILV" : np.array([eep.phasor(8106.92,-87.074),eep.phasor(8106.92,92.926),eep.phasor(1*p,104.036)])},
]

# Evaluate Each Set of Iop and Irt
for n,case in enumerate(CASES):
    IHV = case["IHV"]
    ILV = case["ILV"]
    iop,irt,slp = iopirt(IHV,ILV,HvTAP,LvTAP,corrHV="D-",corrLV="Y",CTRHV=abs(IHV)/5,CTRLV=abs(ILV)/5)
    print("Case",n+1,"Operate Current:",iop,"A")
    print("       Restrain Current:",irt,"A")
    print("       Slope:",np.ceil(slp*100),"%")
```

```
Case 1 Operate Current: [1.3704224 1.3704224 1.3704224] A
       Restrain Current: [3.5680583 3.5680583 3.5680583] A
       Slope: [39. 39. 39.] %
Case 2 Operate Current: [1.3704224 1.3704224 1.3704224] A
       Restrain Current: [3.5680583 3.5680583 3.5680583] A
       Slope: [39. 39. 39.] %
Case 3 Operate Current: [1.5979997  1.51177128 1.3760546 ] A
       Restrain Current: [3.34052197 3.6172429  3.67481904] A
       Slope: [48. 42. 38.] %
Case 4 Operate Current: [2.46783259 2.46783259 2.46924035] A
       Restrain Current: [2.5043133  2.5043133  2.46924035] A
       Slope: [ 99.  99. 100.] %
Case 5 Operate Current: [2.46783259 2.46783259 2.46924035] A
       Restrain Current: [2.5043133  2.5043133  2.46924035] A
       Slope: [ 99.  99. 100.] %
```

Assuming that the relay should trip the breaker for cases 3, 4, and 5, but should remain closed for cases 1 and 2, it is likely that the slope setting of 40% is safe and secure.

Problem 2:

A)

1. CT saturation can impact internal faults by "limiting" a fault current to a specific level so that in protection logic, the bus currents still sum to zero, even if they really don't. This is an extremely unlikely event due to the circumstances that would have to be aligned to support such operation.
2. CT saturation can affect external faults by making them appear (to the relay) as if they are internal. Because the fault current on one or more CTs press the CT into saturation, and the secondary currents don't accurately reflect the primary currents, the relay will perceive the fault as internal since the bus currents will no longer sum to zero. External faults are more of a concern as this behavior will negatively impact protection coordination and potentially cause false trips and incur large expenses for utilities and their customers.
3. A few options will help reduce saturations effects on security.
    - Increasing CT ratings to reduce risk of saturation will certainly have a great benefit.
    - Using coordinated and communications-assited protection may be of benefit. In substations, there are no doubt additional relays monitoring the lines fed by busses, the relays protecting these lines (if not using the same CTs as the bus protection relay) could provide confirmation that validates the current differential issue perceived by the bus protection relay. If the line relay's current doesn't match that of the bus protection relay, there is likely a case of saturation.
    - Reducing CT burden can also significantly improve performance by reducing the likelyhood of CT saturation.
    - High-speed fault identification and supervisory logic can also be used to determine a fault's presence as internal or external and provide supervisory control to block tripping when necessary.

B) Although ring busses may not be quite as simple to apply differential protection, the method may still apply. Since the driving law (KCL) still applies to ring bus configurations, differential protection can still be used, it just may not be as easy to apply. However, in cases where differential protection cannot be applied, breaking bus protection into two or more zones may also be useful. In such cases, the ability to have a dynamically configurable protection relay (like the SEL-487B) is very useful. Such relays can make decisions on the fly to reconfigure zone protection based on breaker status, load status, or other factors. Such performance can improve overall reliablity and security of the system.

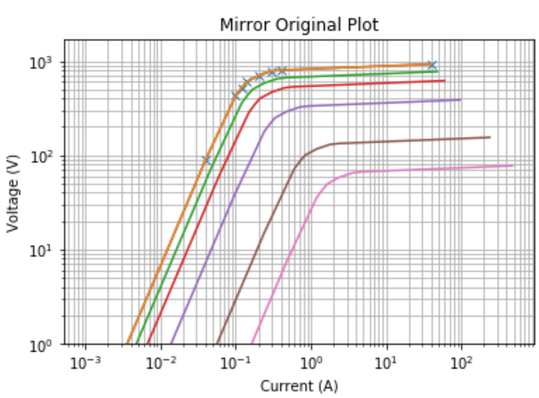
C) Breaker failure is essentially just as the name implies. The breaker fails to either open or close on command. In many cases, this issue is presented as a breaker that fails to close, creating what is known as a "phase-open" case where one of three phases remains open and creates unbalance. Such phenomena on a transmission system can cause extremely adverse conditions by weilding "fault-like" conditions at other protective devices, or conditions that cause a larger system-wide issue, potentially cascading to system instability. Using breaker monitor logic (monitoring the ANSI 52 contact status of the breaker) and sequence logic, relays should accurately be able to detect such failure and either trip as necessary or transfer the trip command (via communications channel) to another protective device whose action is better suited.

D) In combination with differential logic monitoring the current flow through the transformers, contact and status monitoring devices could be implimented to identify the current tap states for the two transformers. Additionally, directional protection supervision could be utilized to aid in identifying where the current is circulating.

Problem 3:

In [8]:

```
# Define Givens:
CTR = 1200/5
CTR_cclass = 600
CTR_RpTurn = 0.0024
Rlead = 0.81
Rstb = 2000
VLL_rat = 230*k
Sbase = 100*M

# Define Fault Characteristics
PHS3 = 45*k
SLG = 40*k
LL = 37*k

# Define CT Data
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]])
TAPS = np.array([240,200,180,160,120,100,80,60,40,20])

# Define Required Functions
def vsetpoint(Rct,k,Rlead,Ifmx,CTR):
    return(1.5*(Rct+k*Rlead)*Ifmx/CTR)
def imin(n,Ie,Irelay,Im,CTR):
    return((n*Ie+Irelay+Im)*CTR)

# Define Curve Functions
vt = lambda N2 : (N2/CTR * excitation[:,[1]]).reshape(9)
Imt = lambda N2 : (CTR/N2 * excitation[:,[0]]).reshape(9)
interpolation = lambda x: np.interp(x,Imt(CTR),vt(CTR))
neginterp = lambda x: np.interp(x,vt(CTR),Imt(CTR))

# Plot All Curves
plt.plot(Imt(CTR),interpolation(Imt(CTR)),'-x')
for i in [1,2,4,6,9,10]:
    tap = TAPS[i-1]
    plt.plot(Imt(tap),vt(tap))
plt.xscale("log")
plt.yscale("log")
plt.ylim((1,12**3))
plt.grid(which="both")
plt.title("Mirror Original Plot")
plt.xlabel("Current (A)")
plt.ylabel("Voltage (V)")
plt.show()
```



In [9]:
```
# A)

# Calculate Setpoint Voltage
CT_R = CTR_RpTurn * CTR
print("CT Resistance:",CT_R,"Ω\n\nA)")
phs3_set = vsetpoint(CT_R,1,Rlead+Rstb,PHS3,CTR)
print("Setpoint Voltage 3-Phase:",phs3_set/k,"kV")
slg_set = vsetpoint(CT_R,2,Rlead+Rstb,SLG,CTR)
print("Setpoint Voltage SLG:",slg_set/k,"kV")
ll_set = vsetpoint(CT_R,2,Rlead+Rstb,LL,CTR)
print("Setpoint Voltage Line-Line:",ll_set/k,"kV\n")

# B)

# Calculate Minimum Primary Internal Fault Current
print("B)")
phs3_min = imin(3,neginterp(phs3_set),phs3_set/Rstb,0,CTR)
print("Minimum Detectable Current (3-Phase):",phs3_min/k,"kA")
slg_min = imin(3,neginterp(slg_set),slg_set/Rstb,0,CTR)
print("Minimum Detectable Current (SLG):",slg_min/k,"kA")
ll_min = imin(3,neginterp(ll_set),ll_set/Rstb,0,CTR)
print("Minimum Detectable Current (Line-Line):",ll_min/k,"kA")

# C)

# Recalculate and Print Results
print("\nC)")
phs3_set = vsetpoint(CT_R,1,Rlead+1200,PHS3,CTR)
print("Setpoint Voltage 3-Phase:",phs3_set/k,"kV")
slg_set = vsetpoint(CT_R,2,Rlead+1200,SLG,CTR)
print("Setpoint Voltage SLG:",slg_set/k,"kV")
ll_set = vsetpoint(CT_R,2,Rlead+1200,LL,CTR)
print("Setpoint Voltage Line-Line:",ll_set/k,"kV\n")
phs3_min = imin(3,neginterp(phs3_set),phs3_set/Rstb,0,CTR)
print("Minimum Detectable Current (3-Phase):",phs3_min/k,"kA")
slg_min = imin(3,neginterp(slg_set),slg_set/Rstb,0,CTR)
print("Minimum Detectable Current (SLG):",slg_min/k,"kA")
ll_min = imin(3,neginterp(ll_set),ll_set/Rstb,0,CTR)
print("Minimum Detectable Current (Line-Line):",ll_min/k,"kA")

# D)

# Recalculate and Print Results
print("\nD)")
phs3_set = vsetpoint(CT_R,1,Rlead+Rstb,50*k,CTR)
print("Setpoint Voltage 3-Phase:",phs3_set/k,"kV")
slg_set = vsetpoint(CT_R,2,Rlead+Rstb,45*k,CTR)
print("Setpoint Voltage SLG:",slg_set/k,"kV")
ll_set = vsetpoint(CT_R,2,Rlead+Rstb,39*k,CTR)
print("Setpoint Voltage Line-Line:",ll_set/k,"kV\n")
phs3_min = imin(4,neginterp(phs3_set),phs3_set/Rstb,0,CTR)
print("Minimum Detectable Current (3-Phase):",phs3_min/k,"kA")
slg_min = imin(4,neginterp(slg_set),slg_set/Rstb,0,CTR)
print("Minimum Detectable Current (SLG):",slg_min/k,"kA")
ll_min = imin(4,neginterp(ll_set),ll_set/Rstb,0,CTR)
print("Minimum Detectable Current (Line-Line):",ll_min/k,"kA")
```

```
CT Resistance: 0.576 Ω

A)
Setpoint Voltage 3-Phase: 562.8898125 kV
Setpoint Voltage SLG: 1000.549 kV
Setpoint Voltage Line-Line: 925.5078249999999 kV

B)
Minimum Detectable Current (3-Phase): 96.3467775 kA
Minimum Detectable Current (SLG): 148.86588 kA
Minimum Detectable Current (Line-Line): 139.86093899999997 kA

C)
Setpoint Voltage 3-Phase: 337.8898125 kV
Setpoint Voltage SLG: 600.549 kV
Setpoint Voltage Line-Line: 555.5078249999999 kV

Minimum Detectable Current (3-Phase): 69.34677750000002 kA
Minimum Detectable Current (SLG): 100.86588 kA
Minimum Detectable Current (Line-Line): 95.46093899999998 kA

D)
Setpoint Voltage 3-Phase: 625.433125 kV
Setpoint Voltage SLG: 1125.617625 kV
```

```
Setpoint Voltage Line-Line: 975.5352750000001 kV

Minimum Detectable Current (3-Phase): 113.451975 kA
Minimum Detectable Current (SLG): 173.474115 kA
Minimum Detectable Current (Line-Line): 155.464233 kA
```

In [ ]:

```

```