Joe Stanley

ECE 525 - HWK 4

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 [40]:

```python
# Define Time-Overcurrent Trip Time Function
def toctriptime(I,Ipickup,TD,curve="U1"):
    """
    toctriptime Function

    Time-OverCurrent Trip Time Calculator, evaluates the time
    to trip for a specific TOC (51) element given the curve
    type, current characteristics and time-dial setting.

    Parameters
    ----------
    I:          float
                Measured Current in Amps
    Ipickup:    float
                Fault Current Pickup Setting (in Amps)
    TD:         float
                Time Dial Setting
    curve:      string, optional
                Name of specified TOC curve, may be entry from set:
                {U1,U2,U3,U4,U5,C1,C2,C3,C4,C5}, default=U1

    Returns
    -------
    tt:         float
                Time-to-Trip for characterized element.
    """
    # Condition Inputs
    curve = curve.upper()
    # Define Dictionary of Constants
    const = {   "U1" : {"A": 0.0104, "B": 0.2256, "P": 0.02},
                "U2" : {"A": 5.95, "B": 0.180, "P": 2.00},
                "U3" : {"A": 3.88, "B": 0.0963, "P": 2.00},
                "U4" : {"A": 5.67, "B": 0.352, "P": 2.00},
                "U5" : {"A": 0.00342, "B": 0.00262, "P": 0.02},
                "C1" : {"A": 0.14, "B":0, "P": 0.02},
                "C2" : {"A": 13.5, "B":0, "P": 2.00},
                "C3" : {"A": 80.0, "B":0, "P": 2.00},
                "C4" : {"A": 120.0, "B":0, "P": 2.00},
                "C5" : {"A": 0.05, "B":0, "P": 0.04}}
    # Load Constants
    A = const[curve]["A"]
    B = const[curve]["B"]
    P = const[curve]["P"]
    # Evaluate M
    M = I / Ipickup
    # Evaluate Trip Time
    tt = TD * (A/(M**P-1)+B)
    return(tt)

# Define Time Overcurrent Reset Time Function
def tocreset(I,Ipickup,TD,curve="U1"):
    """
    tocreset Function

    Function to calculate the time to reset for a TOC
    (Time-OverCurrent, 51) element.
```

```
57
58      Parameters
59      ----------
60      I:          float
61                  Measured Current in Amps
62      Ipickup:    float
63                  Fault Current Pickup Setting (in Amps)
64      TD:         float
65                  Time Dial Setting
66      curve:      string, optional
67                  Name of specified TOC curve, may be entry from set:
68                  {U1,U2,U3,U4,U5,C1,C2,C3,C4,C5}, default=U1
69
70      Returns
71      -------
72      tr:         float
73                  Time-to-Reset for characterized element.
74      """
75      # Condition Inputs
76      curve = curve.upper()
77      # Define Dictionary of Constants
78      C = {   "U1" : 1.08,"U2" : 5.95,"U3" : 3.88,
79              "U4" : 5.67,"U5" : 0.323,"C1" : 13.5,
80              "C2" : 47.3,"C3" : 80.0,"C4" : 120.0,
81              "C5" : 4.85}
82      # Evaluate M
83      M = I / Ipickup
84      # Evaluate Reset Time
85      tr = TD * (C[curve]/(1-M**2))
86      return(tr)
87
88  # Define Pickup Current Calculation
89  def pickup(Iloadmax,Ifaultmin,scale=0,printout=False,units="A"):
90      """
91      pickup Function
92
93      Used to assist in evaluating an optimal phase-over-current pickup
94      setting. Uses maximum load and minimum fault current to provide
95      user assistance.
96
97      Parameters
98      ----------
99      Iloadmax:   float
100                 The maximum load current in amps.
101     Ifaultmin:  float
102                 The minimum fault current in amps.
103     scale:      int, optional
104                 Control scaling to set number of significant figures.
105                 default=0
106     printout:   boolean, optional
107                 Control argument to enable printing of intermediate
108                 stages, default=False.
109     units:      string, optional
110                 String to be appended to any printed output denoting
111                 the units of which are being printed, default="A"
112
113     Returns
```

```
    -------
    setpoint:   float
                The evaluated setpoint at which the function suggests
                the phase-over-current pickup setting be placed.
    """
    IL2 = 2*Iloadmax
    IF2 = Ifaultmin/2
    exponent = len(str(IL2).split('.')[0])
    setpoint = np.ceil(IL2*10**(-exponent+1+scale))*10**(exponent-1-scale)
    if printout:
        print("Range Min:",IL2,units,"\t\tRange Max:",IF2,units)
    if IF2 < setpoint:
        setpoint = IL2
        if IL2 > IF2:
            raise ValueError("Invalid Parameters.")
    if printout:
        print("Current Pickup:",setpoint,units)
    return(setpoint)

# Define Time-Dial Coordination Function
def tdcoordradial(I,CTI,Ipu_up,Ipu_dn,TDdn,curve="U1",scale=1,freq=60):
    """
    tdcoordradial Function

    Function to evaluate the Time-Dial (TD) setting in radial schemes
    where the Coordinating Time Interval (CTI) and the up/downstream
    pickup settings are known along with the TD setting for the
    downstream protection.

    Parameters
    ----------
    I:          float
                Measured fault current in Amps, typically set using the
                maximum fault current available.
    CTI:        float
                Coordinating Time Interval in cycles.
    Ipu_up:     float
                Pickup setting for upstream protection,
                specified in amps
    Ipu_dn:     float
                Pickup setting for downstream protection,
                specified in amps
    TDdn:       float
                Time-Dial setting for downstream protection,
                specified in seconds
    curve:      string, optional
                Name of specified TOC curve, may be entry from set:
                {U1,U2,U3,U4,U5,C1,C2,C3,C4,C5}, default=U1
    scale:      int, optional
                Scaling value used to evaluate a practical TD
                setting, default=1
    freq:       float, optional
                System operating frequency, default=60

    Returns
    -------
    TD:         float
```

```
                Calculated Time-Dial setting according to radial
                scheme logical analysis.
    """
    # Condition Inputs
    curve = curve.upper()
    CTI = CTI/freq # Evaluate in seconds from cycles
    # Define Dictionary of Constants
    const = {   "U1" : {"A": 0.0104, "B": 0.2256, "P": 0.02},
                "U2" : {"A": 5.95, "B": 0.180, "P": 2.00},
                "U3" : {"A": 3.88, "B": 0.0963, "P": 2.00},
                "U4" : {"A": 5.67, "B": 0.352, "P": 2.00},
                "U5" : {"A": 0.00342, "B": 0.00262, "P": 0.02},
                "C1" : {"A": 0.14, "B":0, "P": 0.02},
                "C2" : {"A": 13.5, "B":0, "P": 2.00},
                "C3" : {"A": 80.0, "B":0, "P": 2.00},
                "C4" : {"A": 120.0, "B":0, "P": 2.00},
                "C5" : {"A": 0.05, "B":0, "P": 0.04}}
    # Load Constants
    A = const[curve]["A"]
    B = const[curve]["B"]
    P = const[curve]["P"]
    # Evaluate M
    M = I / Ipu_dn
    # Evaluate Trip Time
    tpu_desired = TDdn * (A/(M**P-1)+B) + CTI
    # Re-Evaluate M
    M = I / Ipu_up
    # Calculate TD setting
    TD = tpu_desired / (A/(M**2-1)+B)
    # Scale and Round
    TD = np.ceil(TD*10**scale)/10**scale
    return(TD)
```

Problem 1

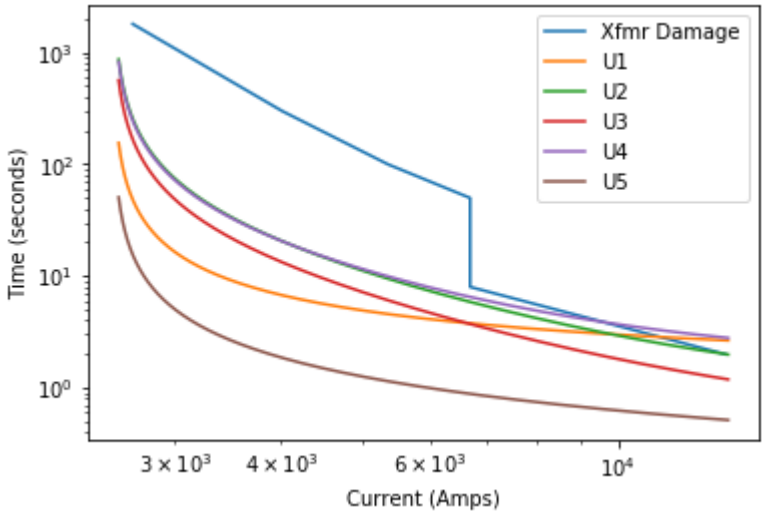
Start by determining the best fit curve associated with breaker B2. We will work from the recloser back, but let us first determine the best curve to use.

A few simplifying assumptions have been made here. It has been assumed that the current drop (due to a fault) down the line increments linearly. Thus the fault current at 60% between two busses will be 40% of the difference in fault currents between those two busses.

In [16]:

```
# Plot Transformer Damage Curve and US-TOC Curves

# Load Transformer Data
xfm_mult = np.array([2,3,4,5,5,10])
xfm_Tlim = np.array([1800,300,100,50,8,2])
# Calculate Rated Transformer Current
Irated = 50*M/(12.47*k*3)
print("Rated Current:",Irated,"A-primary")
xfm_crnt = xfm_mult*Irated

# Plot Transformer Data
plt.plot(xfm_crnt,xfm_Tlim,label="Xfmr Damage")
plt.xscale("log")
plt.yscale("log")

Iload = (3.3*3*M)/(12.47*k*np.sqrt(3)) + 2*3/10*Irated
print("Max Load Current:",Iload,"A-primary")

# Plot TOC Curves
curves = ["U1","U2","U3","U4","U5"]
I = np.arange(min(xfm_crnt)-100,max(xfm_crnt)+100)
Ipickup = pickup(Iload,5728,2,printout=True,units="A-primary")
TD = 5
for curve in curves:
    t = toctriptime(I,Ipickup,TD,curve)
    plt.plot(I,t,label=curve)
plt.legend()
plt.ylabel("Time (seconds)")
plt.xlabel("Current (Amps)")
plt.show()
print("Choose U3 - Very Inverse Curve")
```

```
Rated Current: 1336.5410318096765 A-primary
Max Load Current: 1260.28609984 A-primary
Range Min: 2520.57219968 A-primary                Range Max: 2864.0 A-primary
Current Pickup: 2530.0 A-primary
```




```
Choose U3 - Very Inverse Curve
```

In [17]:

```
# Evaluate CTR For B3 and R1/2

# First determine load
il_b3 = (3.3*3*M)/(12.47*k*np.sqrt(3))
il_r1 = (3.3*2*M)/(12.47*k*np.sqrt(3))
il_r2 = (3.3*M)/(12.47*k*np.sqrt(3))

CTR_b3 = int(np.ceil(il_b3))/5
CTR_r1 = int(np.ceil(il_r1))/5
CTR_r2 = int(np.ceil(il_r2))/5
print("B3 CTR:",CTR_b3)
print("R1 CTR:",CTR_r1)
print("R2 CTR:",CTR_r2)
```

```
B3 CTR: 91.8
R1 CTR: 61.2
R2 CTR: 30.6
```

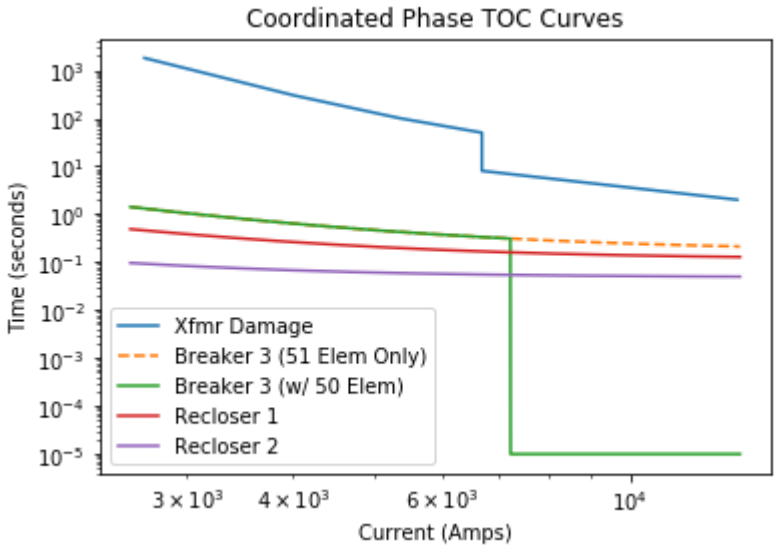
In [18]:

```
# Evaluate the Trip Parameters for Reclosers and Breaker 3

# Current Pickup
R2_pu = pickup(il_r2,5728)
print("R2 Pickup:",R2_pu,"A-primary")
R1_pu = pickup(il_r1,5728)
print("R1 Pickup:",R1_pu,"A-primary")
B3_pu = pickup(il_b3,5728)
print("B3 Pickup:",B3_pu,"A-primary")

# Time Dial Settings
TD_r2 = 0.5
print("R2 Time-Dial:",TD_r2,"sec")
TD_r1 = tdcoordradial(7761.5,6,R1_pu,R2_pu,TD_r2,curve="U3")
print("R1 Time-Dial:",TD_r1,"sec")
TD_b3 = tdcoordradial(9259.8,6,B3_pu,R1_pu,TD_r1,curve="U3")
print("B3 Time-Dial:",TD_b3,"sec")

# Formulate the Instantaneous Element Pickup
# Assume Linear Current Drop Across Line
If_bus3 = 8019.3
If_bus4 = 6682.7
B3_50_pu = np.ceil((If_bus3 - (If_bus3-If_bus4)*0.6)/10)*10
print("B3 50-element Pickup:",B3_50_pu,"A-primary")
```

```
R2 Pickup: 400.0 A-primary
R1 Pickup: 700.0 A-primary
B3 Pickup: 1000.0 A-primary
R2 Time-Dial: 0.5 sec
R1 Time-Dial: 1.2 sec
B3 Time-Dial: 1.8 sec
B3 50-element Pickup: 7220.0 A-primary
```

In [19]:

```
# Plot Trip Curves

# Plot Transformer Data
plt.plot(xfm_crnt,xfm_Tlim,label="Xfmr Damage")
plt.xscale("log")
plt.yscale("log")
# Plot Curves
R2_crv = toctriptime(I,R2_pu,TD_r2,curve="U3")
R1_crv = toctriptime(I,R1_pu,TD_r1,curve="U3")
B3_crv = toctriptime(I,B3_pu,TD_b3,curve="U3")
B3_wo50 = np.copy(B3_crv)
for i, mag in enumerate(I):
    if mag > B3_50_pu:
        B3_crv[i] = 10*u
plt.plot(I,B3_wo50,label="Breaker 3 (51 Elem Only)",linestyle="--")
plt.plot(I,B3_crv,label="Breaker 3 (w/ 50 Elem)")
plt.plot(I,R1_crv,label="Recloser 1")
plt.plot(I,R2_crv,label="Recloser 2")
plt.legend()
plt.ylabel("Time (seconds)")
plt.xlabel("Current (Amps)")
plt.title("Coordinated Phase TOC Curves")
plt.show()
```



Problem 2

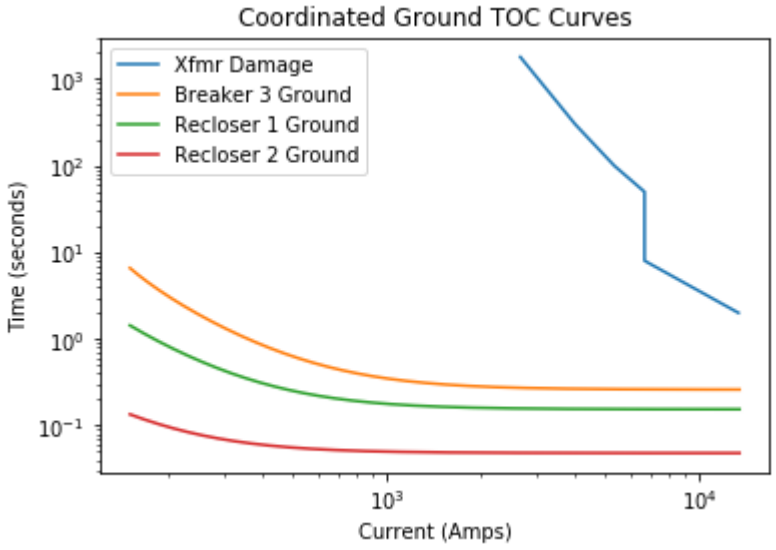
In [20]:

```
# Evaluate Ground Element Settings

# Evaluate Zero-Sequence Current Pickups
I0_bus5 = np.ceil( il_r2 * 0.2 )
I0_bus4 = np.ceil( il_r1 * 0.2 )
I0_bus3 = np.ceil( il_b3 * 0.2 )
print("Recloser 2 Ground Pickup:",I0_bus5,"A-primary")
print("Recloser 1 Ground Pickup:",I0_bus4,"A-primary")
print("Breaker 3 Ground Pickup:",I0_bus3,"A-primary")

# Evaluate Time-Dial Settings
R2_gtd = 0.5
R1_gtd = tdcoordradial(6944.9,6,I0_bus4,I0_bus5,R2_gtd,curve="U3")
B3_gtd = tdcoordradial(9259.8,6,I0_bus3,I0_bus4,R1_gtd,curve="U3")
print("Recloser 2 Ground TD:",R2_gtd,"sec")
print("Recloser 1 Ground TD:",R1_gtd,"sec")
print("Breaker 3 Ground TD:",B3_gtd,"sec")
```

```
Recloser 2 Ground Pickup: 31.0 A-primary
Recloser 1 Ground Pickup: 62.0 A-primary
Breaker 3 Ground Pickup: 92.0 A-primary
Recloser 2 Ground TD: 0.5 sec
Recloser 1 Ground TD: 1.6 sec
Breaker 3 Ground TD: 2.7 sec
```

In [44]:

```
# Plot Trip Curves

# Plot Transformer Data
plt.plot(xfm_crnt,xfm_Tlim,label="Xfmr Damage")
plt.xscale("log")
plt.yscale("log")
# Plot Curves
Ig = np.arange(150,max(I))
R2g_crv = toctriptime(Ig,I0_bus5,R2_gtd,curve="U3")
R1g_crv = toctriptime(Ig,I0_bus4,R1_gtd,curve="U3")
B3g_crv = toctriptime(Ig,I0_bus3,B3_gtd,curve="U3")
plt.plot(Ig,B3g_crv,label="Breaker 3 Ground")
plt.plot(Ig,R1g_crv,label="Recloser 1 Ground")
plt.plot(Ig,R2g_crv,label="Recloser 2 Ground")
plt.legend()
plt.ylabel("Time (seconds)")
plt.xlabel("Current (Amps)")
plt.title("Coordinated Ground TOC Curves")
plt.show()
```



Problem 3

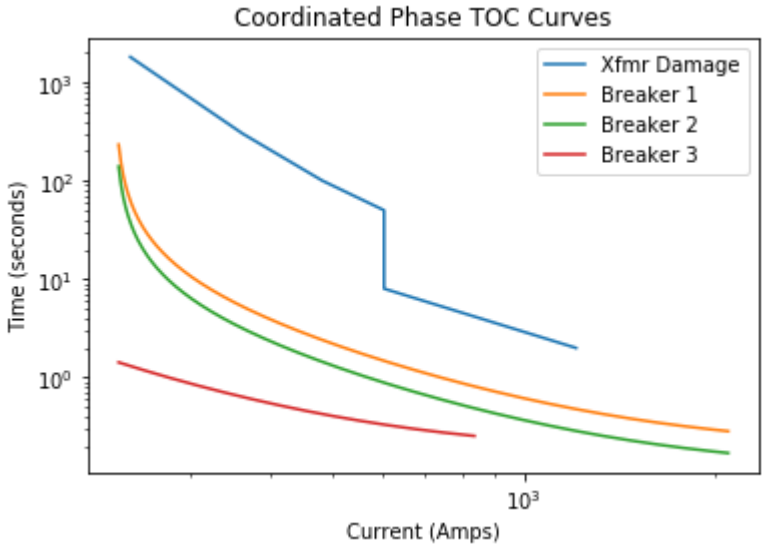
In [41]:

```
# Evaluate the Load Currents

# Begin by evaluating load currents on low-side
Iload = (3.3*3*M)/(12.47*k*np.sqrt(3)) + 2*3/10*Irated
print("Max Load Current:",Iload,"A-primary")
# Convert to High-Side
Iload_prim = 12.47*k/(138*k) * Iload
print("Max Load Current (High-Side):",Iload_prim,"A-prim")

# Evaluate Smallest Phase Fault on High-Side
Ifault_prim = 12.47*k/(138*k) * 5728
print("Smallest Phase Fault (High-Side):",Ifault_prim,"A-prim")

# Evaluate Pickup Setting
B2_pu = pickup(Iload_prim,Ifault_prim,scale=0,printout=False,units="A")
print("Breaker 2 Pickup Setting:",B2_pu,"A-primary")
B1_pu = B2_pu
print("Breaker 1 Pickup Setting:",B1_pu,"A-primary")

# Evaluate Time-Dial Settings
TD_b2 = tdcoordradial(11575*12.47/138,6,B2_pu,B3_pu*12.47/138,TD_b3,curve="U
TD_b1 = tdcoordradial(2091.85,6,B1_pu,B2_pu,TD_b2,curve="U3",scale=1,freq=60
print("Breaker 2 Time-Dial Setting:",TD_b2,"seconds")
print("Breaker 1 Time-Dial Setting:",TD_b1,"seconds")
```

```
Max Load Current: 1260.28609984 A-primary
Max Load Current (High-Side): 113.882374384 A-prim
Smallest Phase Fault (High-Side): 517.5953623188406 A-prim
Breaker 2 Pickup Setting: 227.764748768 A-primary
Breaker 1 Pickup Setting: 227.764748768 A-primary
Breaker 2 Time-Dial Setting: 1.2 seconds
Breaker 1 Time-Dial Setting: 2.0 seconds
```

In [43]:

```
# Plot Trip Curves

# Plot Transformer Data
plt.plot(xfm_crnt*12.47/138,xfm_Tlim,label="Xfmr Damage")
plt.xscale("log")
plt.yscale("log")
# Evaluate Curves
I_prim = np.arange(min(xfm_crnt*12.47/138)-10,2092)
I_sec = np.arange(min(xfm_crnt*12.47/138)-10,9259.8*12.47/138)
B3 = toctriptime(I_sec*138/12.47,B3_pu,TD_b3,curve="U3")
B2 = toctriptime(I_prim,B2_pu,TD_b2,curve="U3")
B1 = toctriptime(I_prim,B1_pu,TD_b1,curve="U3")
plt.plot(I_prim,B1,label="Breaker 1")
plt.plot(I_prim,B2,label="Breaker 2")
plt.plot(I_sec,B3,label="Breaker 3")
plt.legend()
plt.ylabel("Time (seconds)")
plt.xlabel("Current (Amps)")
plt.title("Coordinated Phase TOC Curves")
plt.show()
```



In [ ]:

```
1
```