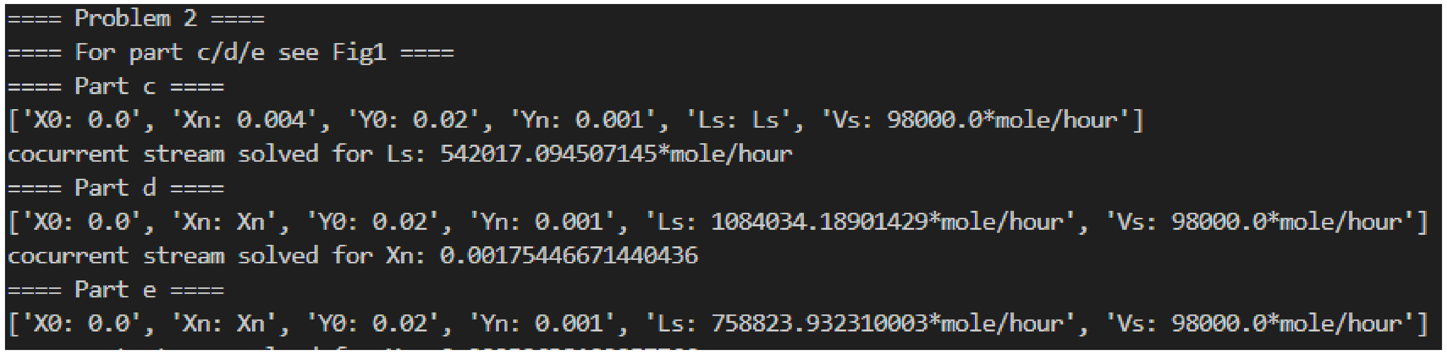
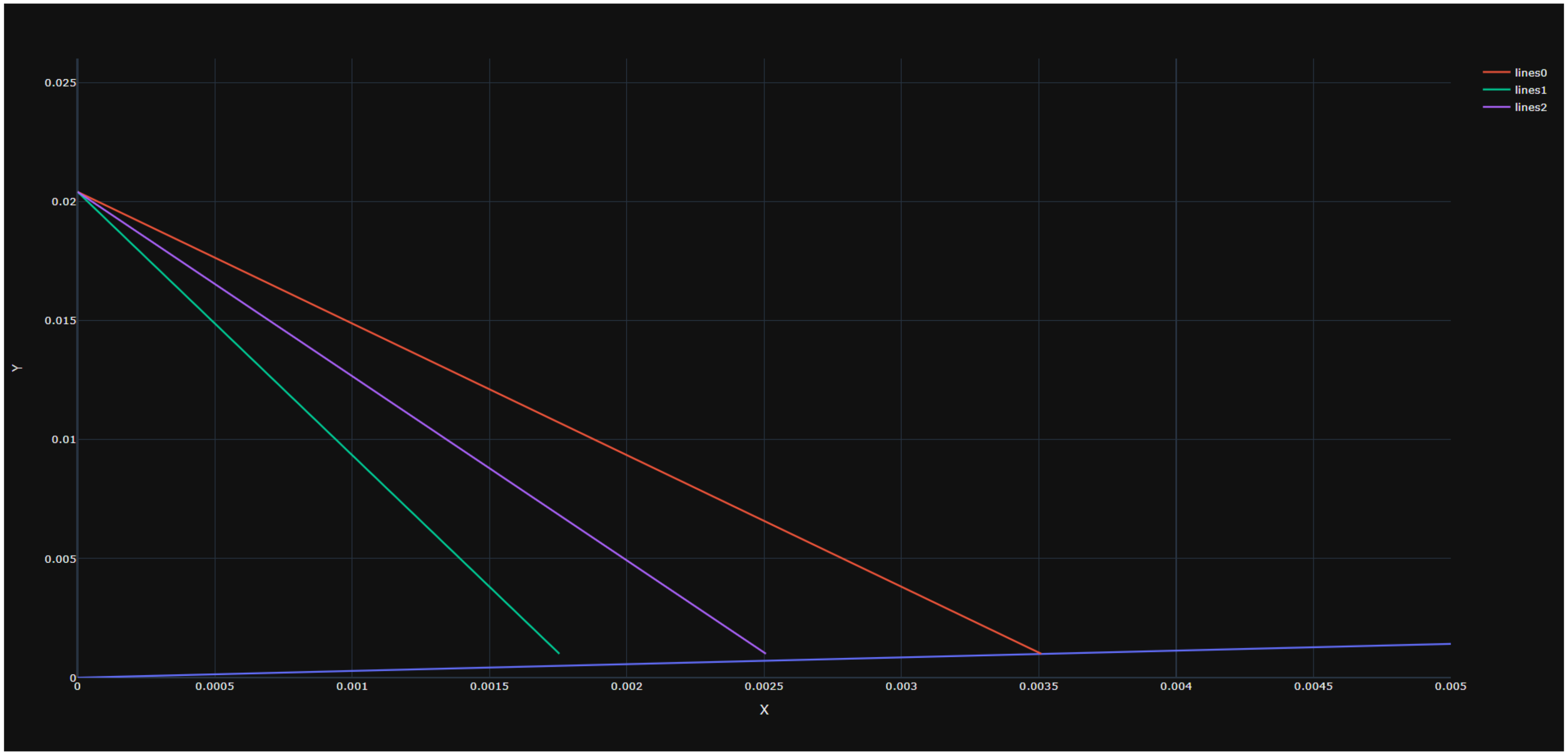
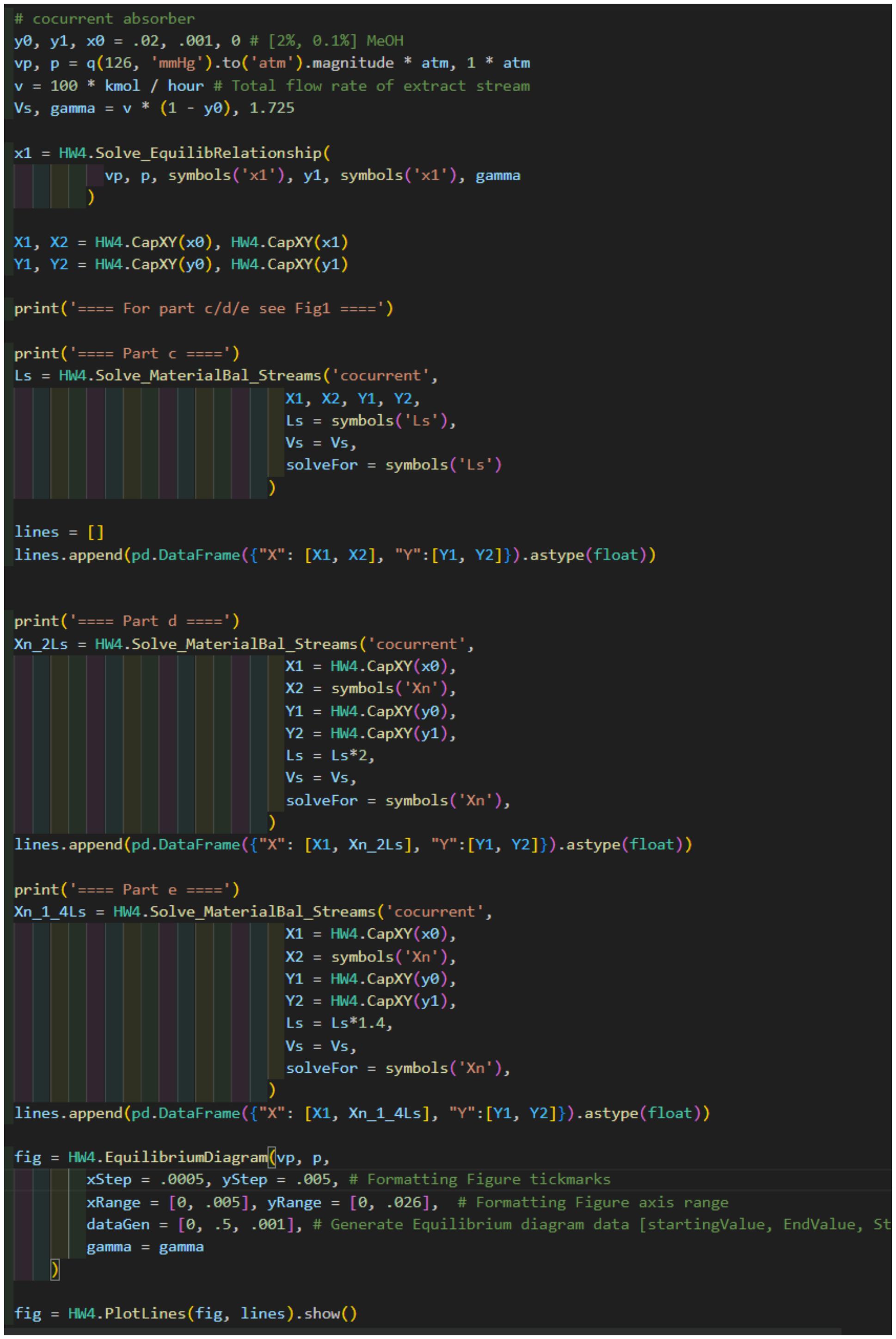
HW4 Problem 2



```
==== Problem 2 ====
==== For part c/d/e see Fig1 ====
==== Part c ====
['X0: 0.0', 'Xn: 0.004', 'Y0: 0.02', 'Yn: 0.001', 'Ls: Ls', 'Vs: 98000.0*mole/hour']
cocurrent stream solved for Ls: 542017.094507145*mole/hour
==== Part d ====
['X0: 0.0', 'Xn: Xn', 'Y0: 0.02', 'Yn: 0.001', 'Ls: 1084034.18901429*mole/hour', 'Vs: 98000.0*mole/hour']
cocurrent stream solved for Xn: 0.00175446671440436
==== Part e ====
['X0: 0.0', 'Xn: Xn', 'Y0: 0.02', 'Yn: 0.001', 'Ls: 758823.932310003*mole/hour', 'Vs: 98000.0*mole/hour']
```

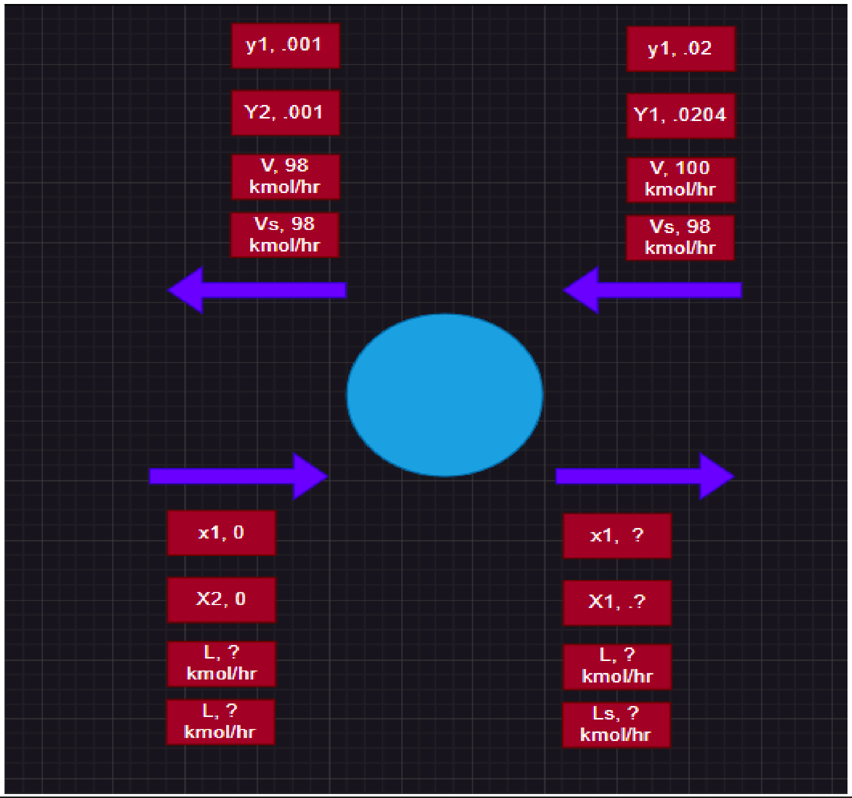


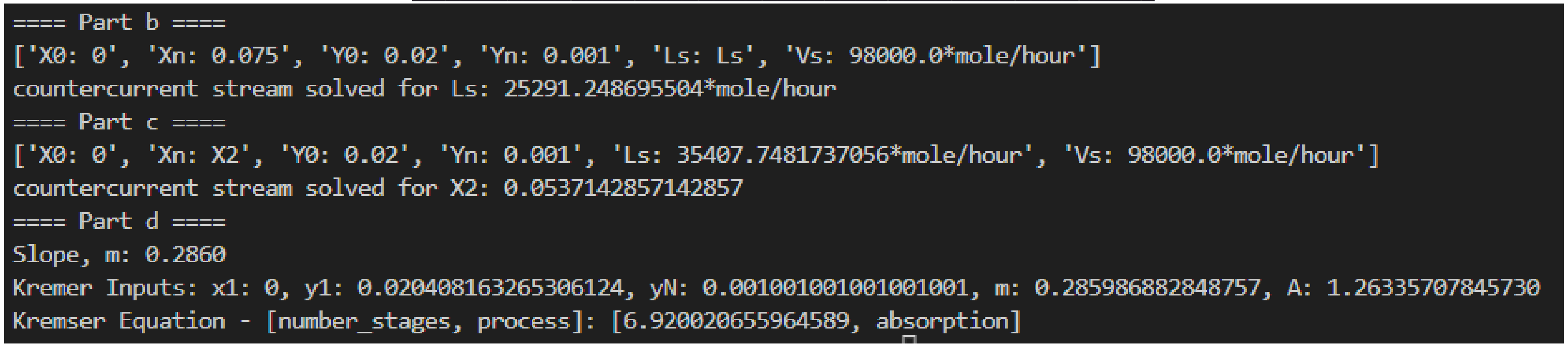
P



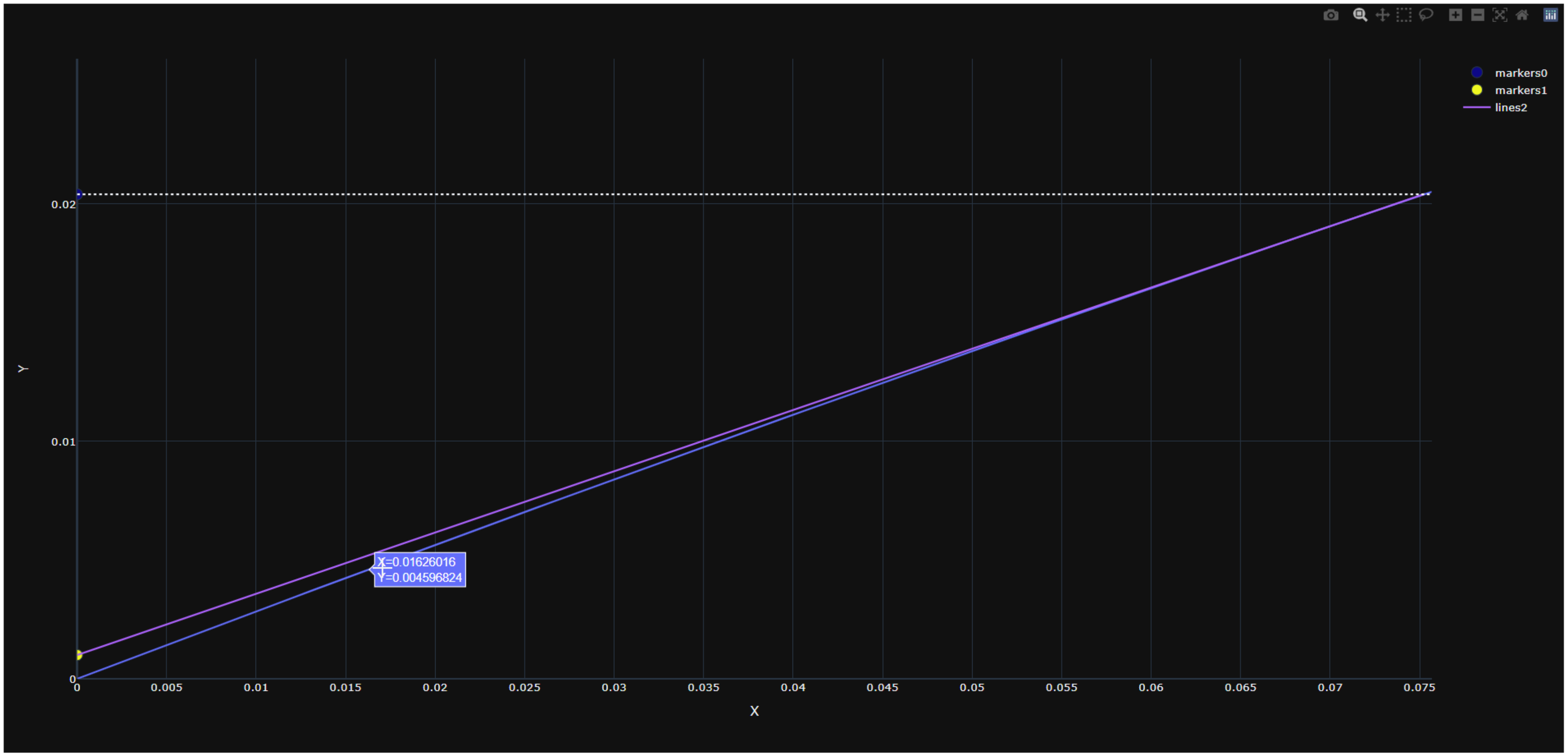
```python
# cocurrent absorber
y0, y1, x0 = .02, .001, 0 # [2%, 0.1%] MeOH
vp, p = q(126, 'mmHg').to('atm').magnitude * atm, 1 * atm
v = 100 * kmol / hour # Total flow rate of extract stream
Vs, gamma = v * (1 - y0), 1.725

x1 = HW4.Solve_EquilibRelationship(
        vp, p, symbols('x1'), y1, symbols('x1'), gamma
        )

X1, X2 = HW4.CapXY(x0), HW4.CapXY(x1)
Y1, Y2 = HW4.CapXY(y0), HW4.CapXY(y1)

print('==== For part c/d/e see Fig1 ====')

print('==== Part c ====')
Ls = HW4.Solve_MaterialBal_Streams('cocurrent',
                              X1, X2, Y1, Y2,
                              Ls = symbols('Ls'),
                              Vs = Vs,
                              solveFor = symbols('Ls')
                             )

lines = []
lines.append(pd.DataFrame({"X": [X1, X2], "Y":[Y1, Y2]}).astype(float))


print('==== Part d ====')
Xn_2Ls = HW4.Solve_MaterialBal_Streams('cocurrent',
                              X1 = HW4.CapXY(x0),
                              X2 = symbols('Xn'),
                              Y1 = HW4.CapXY(y0),
                              Y2 = HW4.CapXY(y1),
                              Ls = Ls*2,
                              Vs = Vs,
                              solveFor = symbols('Xn'),
                             )
lines.append(pd.DataFrame({"X": [X1, Xn_2Ls], "Y":[Y1, Y2]}).astype(float))

print('==== Part e ====')
Xn_1_4Ls = HW4.Solve_MaterialBal_Streams('cocurrent',
                              X1 = HW4.CapXY(x0),
                              X2 = symbols('Xn'),
                              Y1 = HW4.CapXY(y0),
                              Y2 = HW4.CapXY(y1),
                              Ls = Ls*1.4,
                              Vs = Vs,
                              solveFor = symbols('Xn'),
                             )
lines.append(pd.DataFrame({"X": [X1, Xn_1_4Ls], "Y":[Y1, Y2]}).astype(float))

fig = HW4.EquilibriumDiagram(vp, p,
        xStep = .0005, yStep = .005, # Formatting Figure tickmarks
        xRange = [0, .005], yRange = [0, .026],  # Formatting Figure axis range
        dataGen = [0, .5, .001], # Generate Equilibrium diagram data [startingValue, EndValue, St
        gamma = gamma
    )

fig = HW4.PlotLines(fig, lines).show()
```

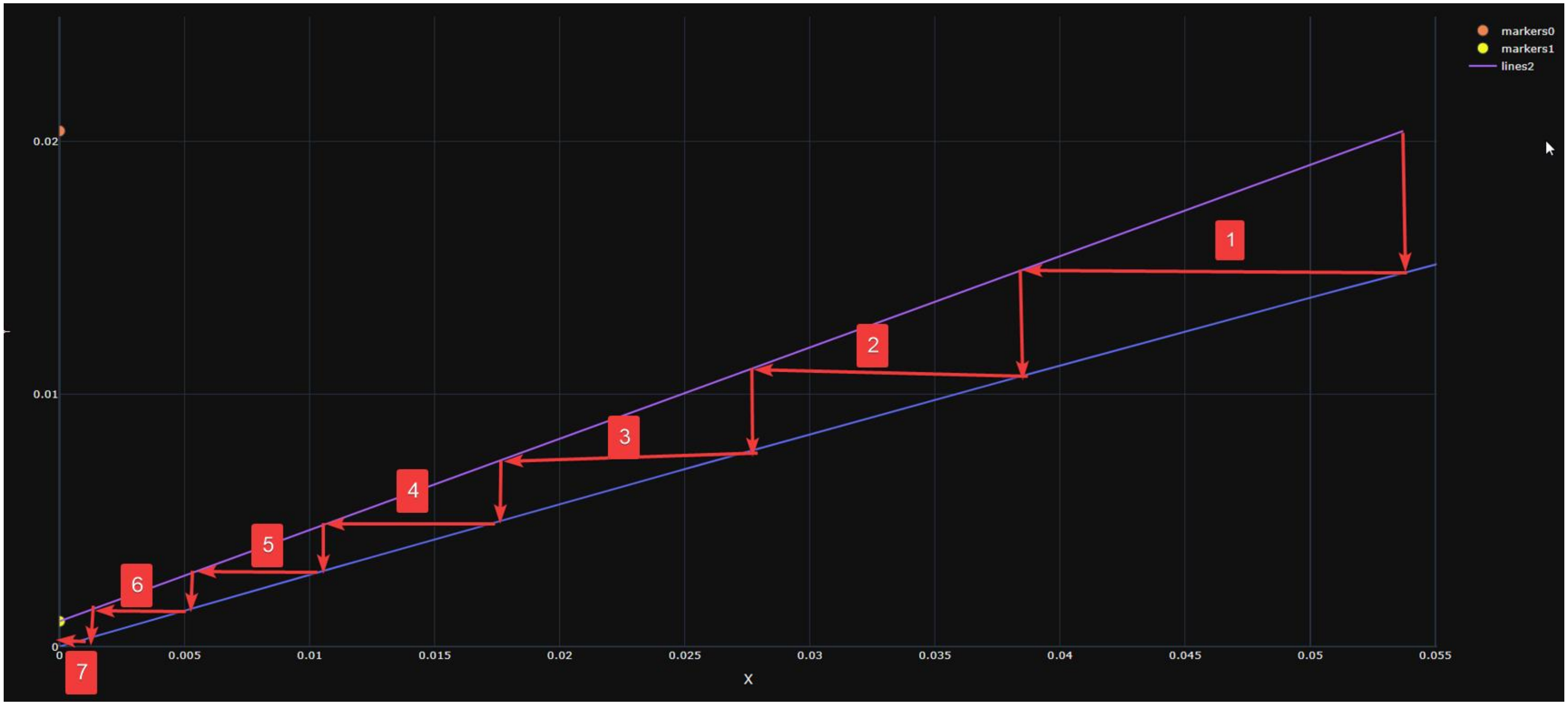
HW5 Problem 1



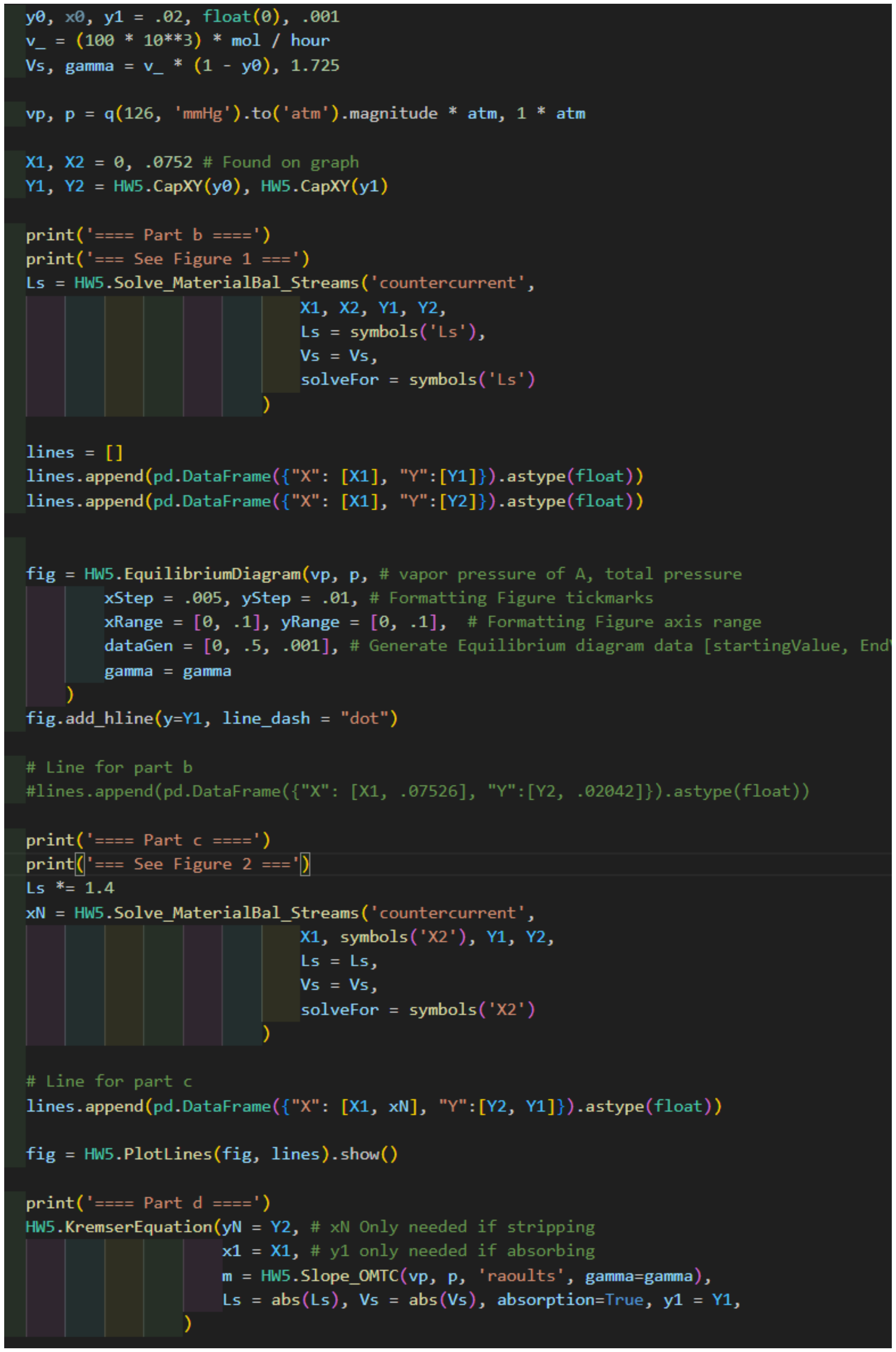


```
==== Part b ====
['X0: 0', 'Xn: 0.075', 'Y0: 0.02', 'Yn: 0.001', 'Ls: Ls', 'Vs: 98000.0*mole/hour']
countercurrent stream solved for Ls: 25291.248695504*mole/hour
==== Part c ====
['X0: 0', 'Xn: X2', 'Y0: 0.02', 'Yn: 0.001', 'Ls: 35407.7481737056*mole/hour', 'Vs: 98000.0*mole/hour']
countercurrent stream solved for X2: 0.0537142857142857
==== Part d ====
Slope, m: 0.2860
Kremer Inputs: x1: 0, y1: 0.020408163265306124, yN: 0.001001001001001001, m: 0.285986882848757, A: 1.26335707845730
Kremser Equation - [number_stages, process]: [6.920020655964589, absorption]
```


markers0
markers1
lines2
X=0.01626016
Y=0.004596824
0.02
0.01
0
Y
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
0.055
0.06
0.065
0.07
0.075
X

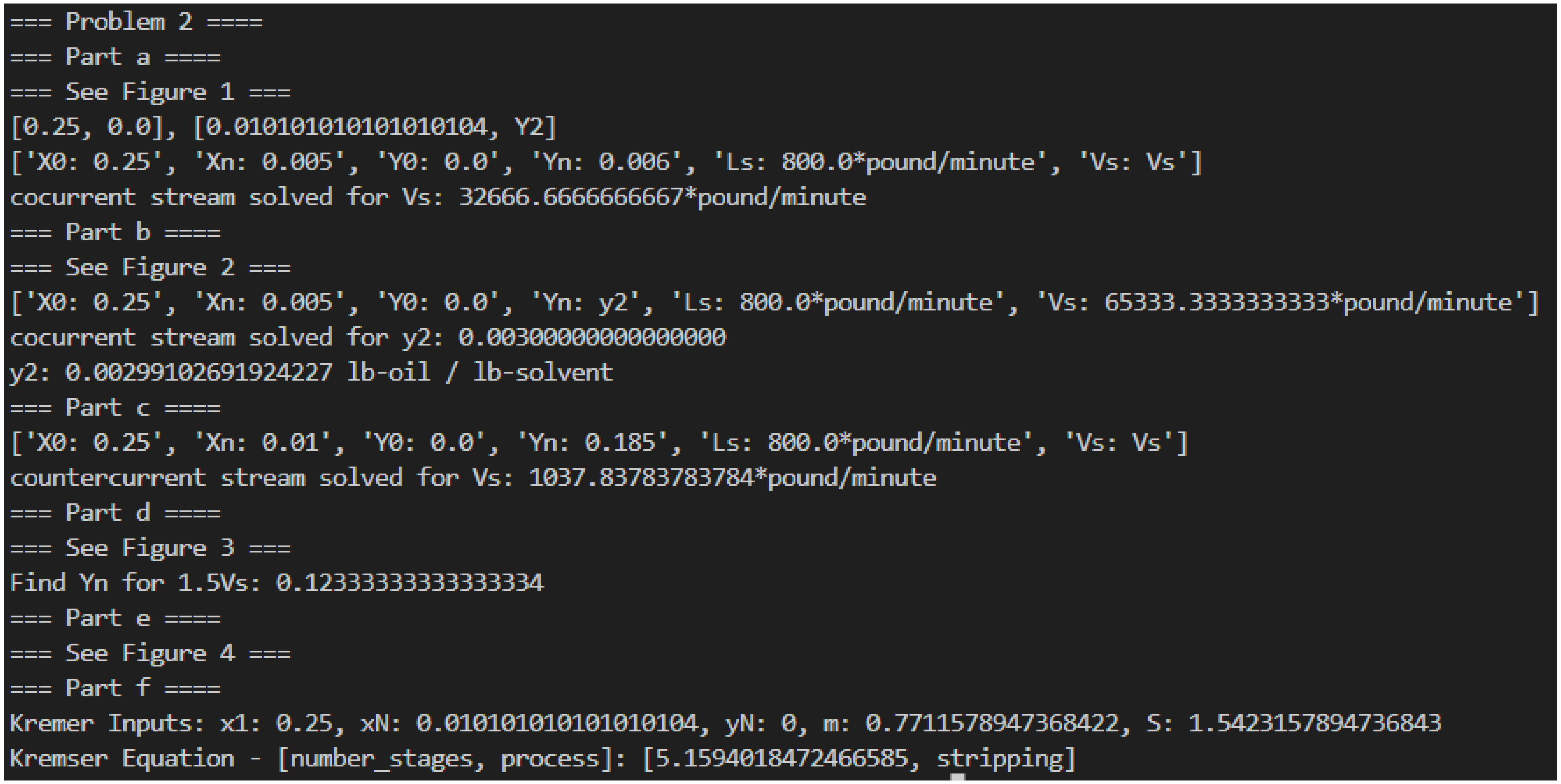


markers0
markers1
lines2
0.02
0.01
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
0.055
x
1
2
3
4
5
6
7



```python
y0, x0, y1 = .02, float(0), .001
v_ = (100 * 10**3) * mol / hour
Vs, gamma = v_ * (1 - y0), 1.725

vp, p = q(126, 'mmHg').to('atm').magnitude * atm, 1 * atm

X1, X2 = 0, .0752 # Found on graph
Y1, Y2 = HW5.CapXY(y0), HW5.CapXY(y1)

print('==== Part b ====')
print('=== See Figure 1 ===')
Ls = HW5.Solve_MaterialBal_Streams('countercurrent',
                            X1, X2, Y1, Y2,
                            Ls = symbols('Ls'),
                            Vs = Vs,
                            solveFor = symbols('Ls')
                        )

lines = []
lines.append(pd.DataFrame({"X": [X1], "Y":[Y1]}).astype(float))
lines.append(pd.DataFrame({"X": [X1], "Y":[Y2]}).astype(float))


fig = HW5.EquilibriumDiagram(vp, p, # vapor pressure of A, total pressure
        xStep = .005, yStep = .01, # Formatting Figure tickmarks
        xRange = [0, .1], yRange = [0, .1],  # Formatting Figure axis range
        dataGen = [0, .5, .001], # Generate Equilibrium diagram data [startingValue, End
        gamma = gamma
    )
fig.add_hline(y=Y1, line_dash = "dot")

# Line for part b
#lines.append(pd.DataFrame({"X": [X1, .07526], "Y":[Y2, .02042]}).astype(float))

print('==== Part c ====')
print('=== See Figure 2 ===')
Ls *= 1.4
xN = HW5.Solve_MaterialBal_Streams('countercurrent',
                            X1, symbols('X2'), Y1, Y2,
                            Ls = Ls,
                            Vs = Vs,
                            solveFor = symbols('X2')
                        )

# Line for part c
lines.append(pd.DataFrame({"X": [X1, xN], "Y":[Y2, Y1]}).astype(float))

fig = HW5.PlotLines(fig, lines).show()

print('==== Part d ====')
HW5.KremserEquation(yN = Y2, # xN Only needed if stripping
                    x1 = X1, # y1 only needed if absorbing
                    m = HW5.Slope_OMTC(vp, p, 'raoults', gamma=gamma),
                    Ls = abs(Ls), Vs = abs(Vs), absorption=True, y1 = Y1,
                )
```

Problem 2



```
=== Problem 2 ====
=== Part a ====
=== See Figure 1 ===
[0.25, 0.0], [0.010101010101010104, Y2]
['X0: 0.25', 'Xn: 0.005', 'Y0: 0.0', 'Yn: 0.006', 'Ls: 800.0*pound/minute', 'Vs: Vs']
cocurrent stream solved for Vs: 32666.6666666667*pound/minute
=== Part b ====
=== See Figure 2 ===
['X0: 0.25', 'Xn: 0.005', 'Y0: 0.0', 'Yn: y2', 'Ls: 800.0*pound/minute', 'Vs: 65333.3333333333*pound/minute']
cocurrent stream solved for y2: 0.00300000000000000
y2: 0.00299102691924227 lb-oil / lb-solvent
=== Part c ====
['X0: 0.25', 'Xn: 0.01', 'Y0: 0.0', 'Yn: 0.185', 'Ls: 800.0*pound/minute', 'Vs: Vs']
countercurrent stream solved for Vs: 1037.83783783784*pound/minute
=== Part d ====
=== See Figure 3 ===
Find Yn for 1.5Vs: 0.12333333333333334
=== Part e ====
=== See Figure 4 ===
=== Part f ====
Kremer Inputs: x1: 0.25, xN: 0.010101010101010104, yN: 0, m: 0.7711578947368422, S: 1.5423157894736843
Kremser Equation - [number_stages, process]: [5.1594018472466585, stripping]
```

Fig 1

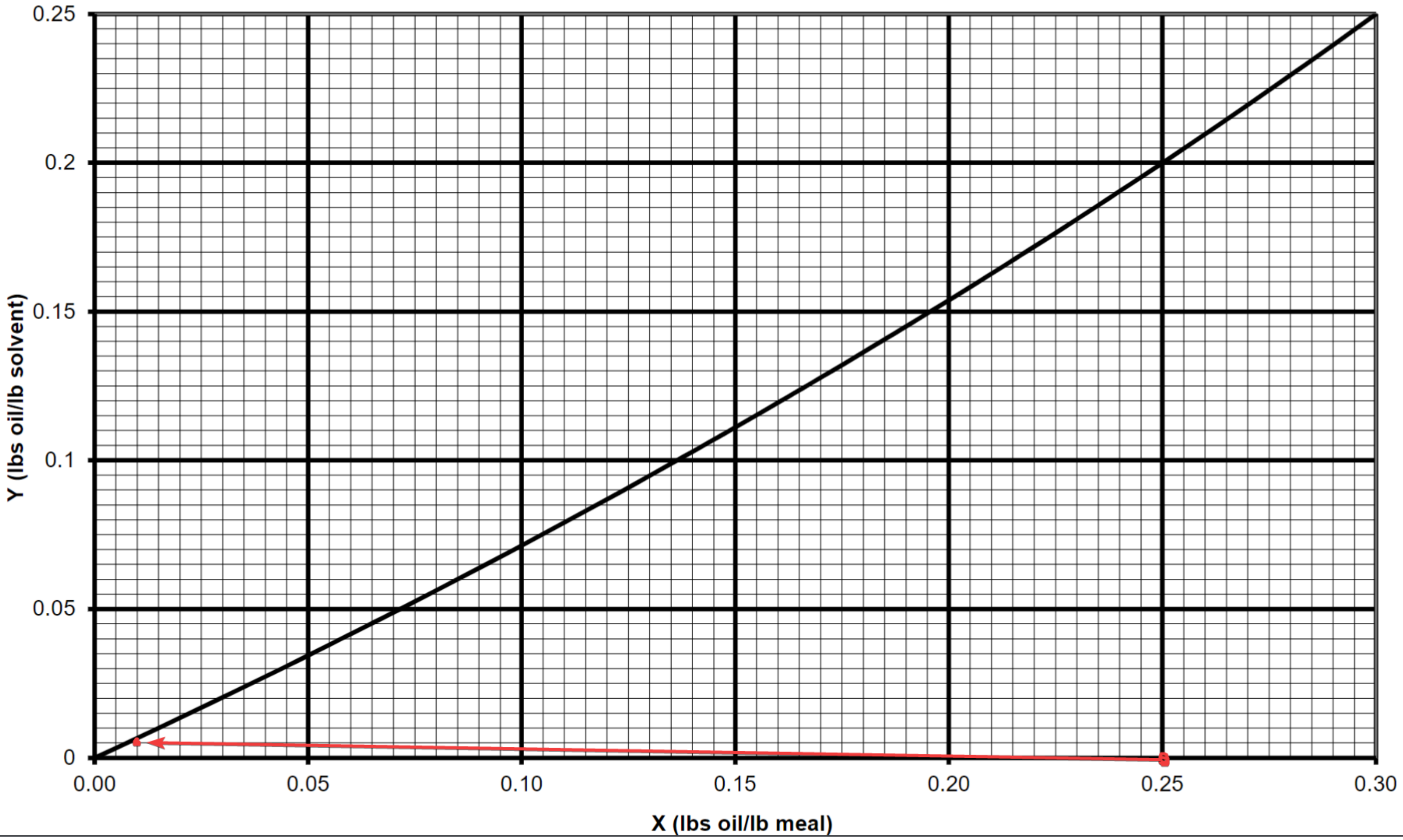
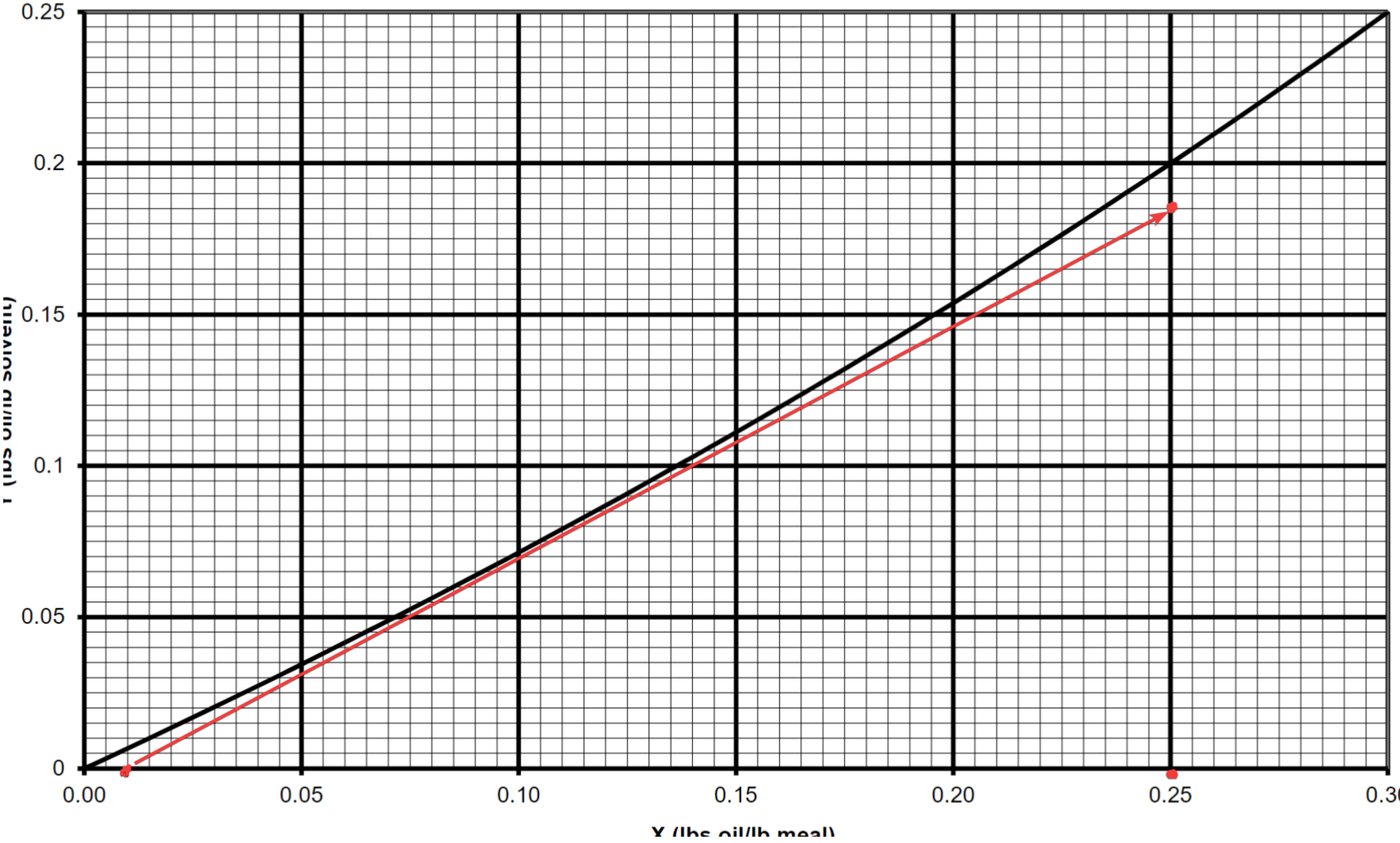
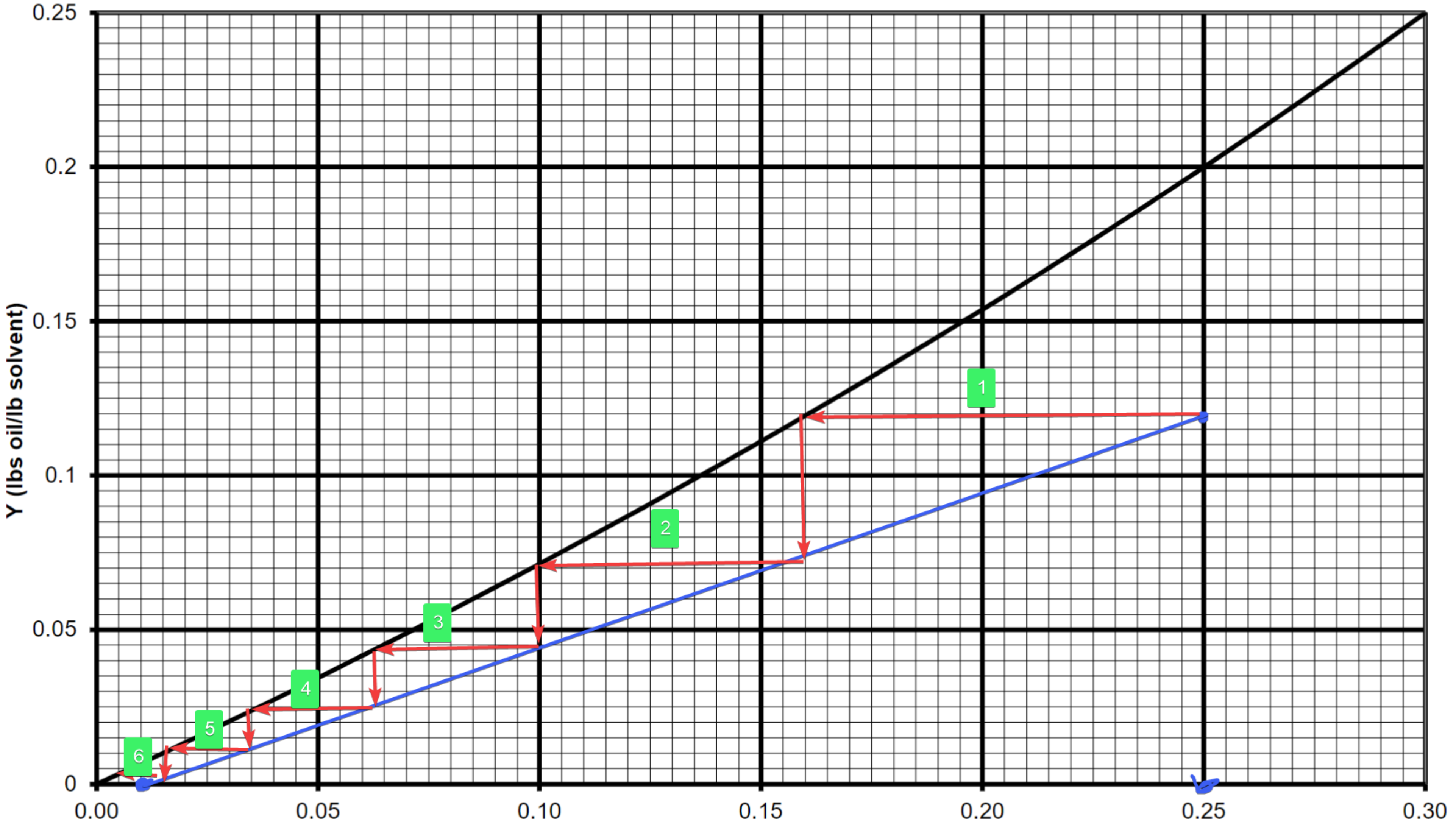

0.25
0.2
0.15
0.1
0.05
0
Y (lbs oil/lb solvent)
0.00
0.05
0.10
0.15
0.20
0.25
0.30
X (lbs oil/lb meal)

Fig 2



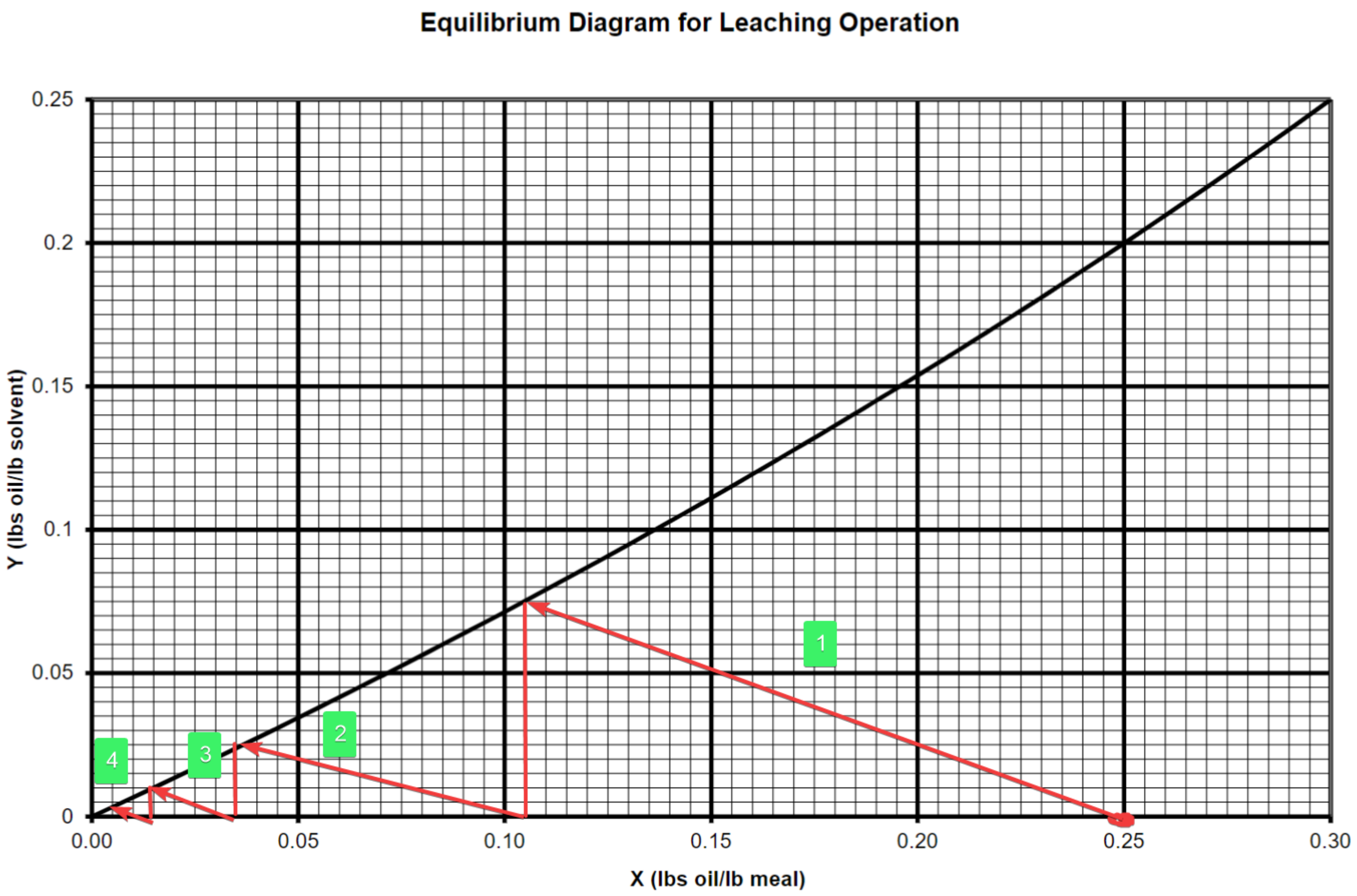
0.25
0.2
0.15
0.1
0.05
0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Y (lbs oil/lb solvent)
X (lbs oil/lb meal)

Fig 3

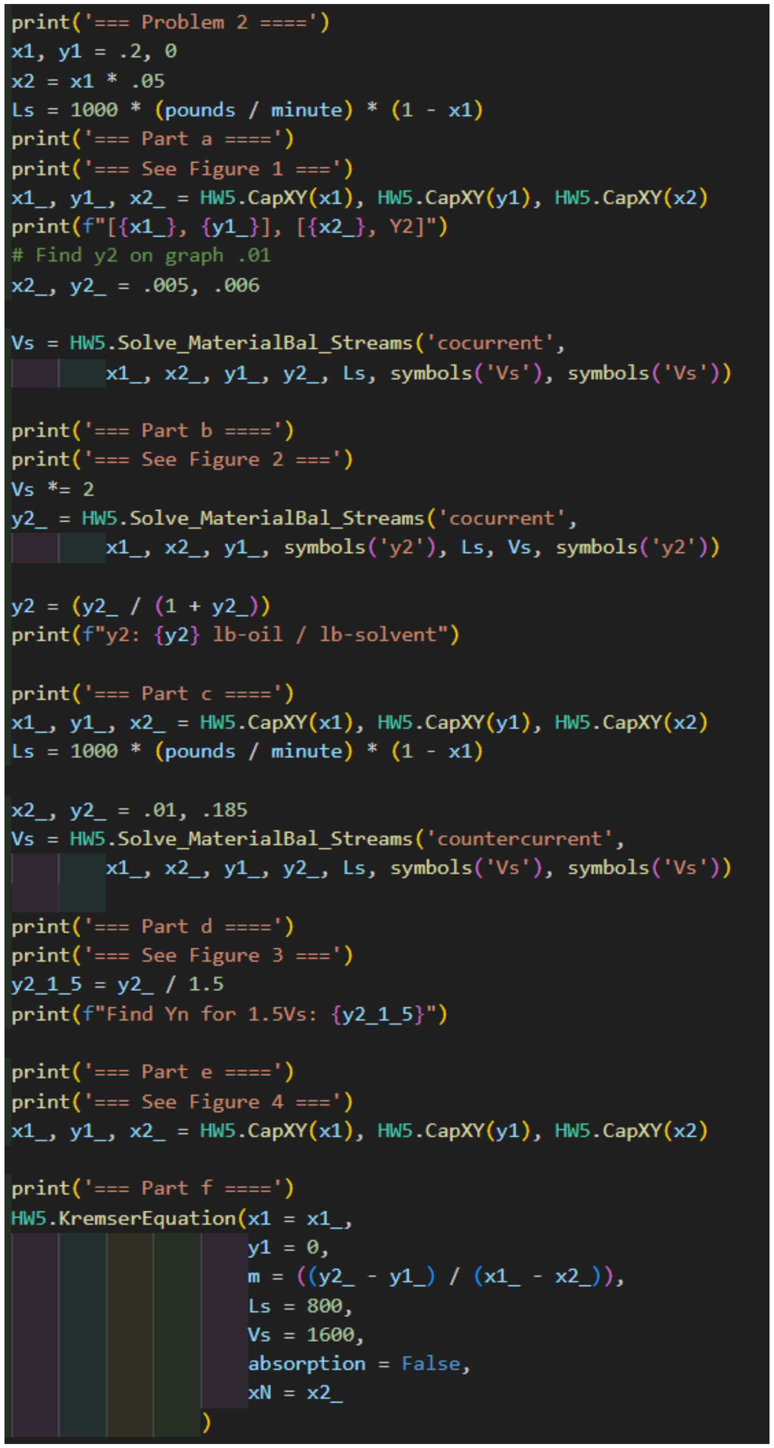


Y (lbs oil/lb solvent)
0.25
0.2
0.15
0.1
0.05
0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
1
2
3
4
5
6

Fig4



Equilibrium Diagram for Leaching Operation
Y (lbs oil/lb solvent)
X (lbs oil/lb meal)
0.25
0.2
0.15
0.1
0.05
0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
1
2
3
4



```python
print('=== Problem 2 ====')
x1, y1 = .2, 0
x2 = x1 * .05
Ls = 1000 * (pounds / minute) * (1 - x1)
print('=== Part a ====')
print('=== See Figure 1 ===')
x1_, y1_, x2_ = HW5.CapXY(x1), HW5.CapXY(y1), HW5.CapXY(x2)
print(f"[{x1_}, {y1_}], [{x2_}, Y2]")
# Find y2 on graph .01
x2_, y2_ = .005, .006

Vs = HW5.Solve_MaterialBal_Streams('cocurrent',
        x1_, x2_, y1_, y2_, Ls, symbols('Vs'), symbols('Vs'))

print('=== Part b ====')
print('=== See Figure 2 ===')
Vs *= 2
y2_ = HW5.Solve_MaterialBal_Streams('cocurrent',
        x1_, x2_, y1_, symbols('y2'), Ls, Vs, symbols('y2'))

y2 = (y2_ / (1 + y2_))
print(f"y2: {y2} lb-oil / lb-solvent")

print('=== Part c ====')
x1_, y1_, x2_ = HW5.CapXY(x1), HW5.CapXY(y1), HW5.CapXY(x2)
Ls = 1000 * (pounds / minute) * (1 - x1)

x2_, y2_ = .01, .185
Vs = HW5.Solve_MaterialBal_Streams('countercurrent',
        x1_, x2_, y1_, y2_, Ls, symbols('Vs'), symbols('Vs'))

print('=== Part d ====')
print('=== See Figure 3 ===')
y2_1_5 = y2_ / 1.5
print(f"Find Yn for 1.5Vs: {y2_1_5}")

print('=== Part e ====')
print('=== See Figure 4 ===')
x1_, y1_, x2_ = HW5.CapXY(x1), HW5.CapXY(y1), HW5.CapXY(x2)

print('=== Part f ====')
HW5.KremserEquation(x1 = x1_,
                    y1 = 0,
                    m = ((y2_ - y1_) / (x1_ - x2_)),
                    Ls = 800,
                    Vs = 1600,
                    absorption = False,
                    xN = x2_
                )
```

Source Code: https://github.com/hunterviolette/CHE362

Methods:
https://github.com/hunterviolette/CHE362/blob/master/scripts/baseFunctions.py

File:

https://github.com/hunterviolette/CHE362/blob/master/scripts/hw5.py