# Capstone 3

December 6, 2019

# 1 Capstone 3 - Distillation

## 1.1 Group 7

**December $3^{rd}$, 2019**

### 1.1.1 Group Members

J. Alex Cole Natasha Andrews Michael Garvey

```
In [1]: # Required Packages
        import matplotlib.pyplot as plt
        import numpy as np
```

### 1.1.2 Definition and Approach

The problem is to construct a distillation column Mcabe-Thiele diagram with reflux and reboil optimization for a selected mass flow rate of the inlet, boiler, and condenser inside of the framework of python 3. The group is to also find the $V_B : V_{B,min}$ ratio through hand calculations.

Provide the code to test the problem with the input paramters below.

| input parameters | variable name | value |
|---|---|---|
| Relative Volatility | $\alpha^*_{BA}$ | 3.0 |
| Margules Parameter | $A$ | -0.9 |
| Azetrope Concentration | $x_{B,az}$ | 0.765 |
| Feed Composition | $z_{B,F}$ | 0.5 |
| Distillate Composition | $x_{B,D}$ | 0.735 |
| Boiler Composition | $x_{B,B}$ | 0.05 |
| Optimal Reflux Ratio | $R/R_{\min}$ | 1.5 |
| Vapor Quality | $q$ | 0.3 |

In order to solve more the majority of the functions a root solving method is used. In this case the bisection method.

```
In [2]: from general_purpose import bisection
```

$$y(x) = x$$

```
In [3]: def create_yxLine(x):
            return x
```

$$y(x_B) = \frac{\alpha_{BA} x}{x\alpha_{BA} + \frac{\gamma_B}{\gamma_A}(1-x)}$$

$$\gamma_B = 10^{A(1-x)^2}$$

$$\gamma_A = 10^{Ax^2}$$

```
In [4]: def create_EQLine(relative_volatility=1, margules_parameter=1):
            gamma_b = lambda x: 10 ** (margules_parameter * (1 - x) ** 2)
            gamma_a = lambda x: 10 ** (margules_parameter * x ** 2)
            return lambda x: (relative_volatility * x) / (x * relative_volatility + (gamma_b(x
```

$$y(x) = \frac{q}{q-1}x - \frac{z_F}{q-1}$$

```
In [5]: def create_qLine(feed_composition, vapor_quality):
            return np.poly1d([(vapor_quality / (vapor_quality - 1)), -1 * feed_composition / (v
```

Solving for the intersection of vapor quality line and the equlibrium line using root finding to solve

$$\frac{\alpha_{AB} x}{x\alpha_{BA} + \frac{\gamma_B}{\gamma_A}(1-x)} = \frac{q}{q-1}x - \frac{z_F}{q-1}$$

```
In [6]: def calc_zFEQ(relative_volatility, margules_parameter, zF, vapor_quality):
            equilibrium_line = create_EQLine(relative_volatility, margules_parameter)
            vapor_quality_line = create_qLine(zF, vapor_quality)

            # This needs to be replaced with fsolve instead of bisection method
            # Finding intersection of vapor quality line and equilibrium line in terms of x co
            g = lambda x: equilibrium_line(x) - vapor_quality_line(x)
            intersect_eq_q = bisection(g)
            return [intersect_eq_q, equilibrium_line(intersect_eq_q)]
```

$$R_{min} = \frac{(L/V)_{min}}{1-(L/V)_{min}}$$

```
In [7]: def calc_Rmin(alpha_ba, A, xD, zF, q):
            # reflux_minimum = (L / V) / (1 - L / V)
            q_line = create_qLine(zF, q)
            equilibrium_line = create_EQLine(alpha_ba, A)
            # This needs to be replaced with fsolve instead of bisection method
            # Finding intersection of vapor quality line and equilibrium line in terms of x co
            g = lambda x: equilibrium_line(x) - q_line(x)
            intersect_eq_q = bisection(g)
```

```python
    # Find the value of the reflux ratio from the minimum reflux ratio
    m = (xD - q_line(intersect_eq_q)) / (xD - intersect_eq_q)
    reflux_minimum = m / (1 - m)
    return reflux_minimum
```

$$y(x) = \frac{V_B + 1}{V_B} x - \frac{x_B}{V_B}$$

In [8]:
```python
def calc_VB_min(alpha_ba, A, xB, zF, q):
    q_line = create_qLine(zF, q)
    equilibrium_line = create_EQLine(alpha_ba, A)
    # This needs to be replaced with fsolve instead of bisection method
    # Finding intersection of vapor quality line and equilibrium line in terms of x co
    g = lambda x: equilibrium_line(x) - q_line(x)
    intersect_eq_q = bisection(g)
    # Find the value of the reflux ratio from the minimum reflux ratio
    m = (xB - q_line(intersect_eq_q)) / (xB - intersect_eq_q)
    VB_min = 1 / (m - 1)
    return VB_min
```

In [9]:
```python
def calc_VVminBminOp(alpha_ba, A, xB, xD, zF, q, R_R_min):
    """returns the V/VBmin ratio at operation"""
    equilibrium_line = create_EQLine(alpha_ba, A)
    vapor_quality_line = create_qLine(zF, q)

    # This needs to be replaced with fsolve instead of bisection method
    # Finding intersection of vapor quality line and equilibrium line in terms of x co
    g = lambda x: equilibrium_line(x) - vapor_quality_line(x)
    intersect_eq_q = bisection(g)
    # Find the value of the reflux ratio from the minimum reflux ratio
    m = (xD - vapor_quality_line(intersect_eq_q)) / (xD - intersect_eq_q)
    R_min = m / (1 - m)
    reflux_ratio = R_min * R_R_min
    reflux_line = create_RLine(xD, reflux_ratio)

    # Root solving for the Reflux line and vapor quality line
    reflux_quality_subtraction = lambda x: reflux_line(x) - vapor_quality_line(x)
    intersect_q_R = bisection(reflux_quality_subtraction)

    # Next Step is to VB_min
    VB_min = calc_VB_min(alpha_ba, A, xB, zF, q)

    # Calculate VB value
    VB_slope = (vapor_quality_line(intersect_q_R) - xB) / (intersect_q_R - xB)
    VB = 1 / (VB_slope - 1)
    return VB / VB_min
```

$$y(x_B) = \frac{R}{R+1} x_B + \frac{x_D}{R+1}$$

In [10]:
```
def create_RLine(distillate_composition, reflux_ratio):
    return np.poly1d([reflux_ratio / (reflux_ratio + 1), distillate_composition / (re
```

$$(V_B)_{min} = \frac{1}{(\bar{L}/\bar{V}) - 1}$$

In [11]:
```
def create_VBLine(boiler_composition, reboil_ratio):
    return np.poly1d([(reboil_ratio + 1) / reboil_ratio, -1 * boiler_composition / re
```

**Bonus** Creating a function that can display the minimum number of trays needed in order to perform the distillation

In [12]:
```
def create_steps(equilibrium_line, VB_line, q_line, reflux_line, xB, xD):
    done = False
    step = 0
    point_a_x = [xD]
    point_a_y = [create_yxLine(xD)]
    while not done:
        if step % 2 == 0:
            """Creating horizontal lines"""
            point_a_y.append(point_a_y[-1])
            equilibrium_intercept = lambda x: equilibrium_line(x) - point_a_y[-1]
            point_a_x.append(bisection(equilibrium_intercept))

        else:
            if point_a_y[-1] > q_line(point_a_x[-1]):
                """If reflux_line(x) point is above the vapor quality line,
                then find the y value of the intercept of the reflux line"""
                point_a_x.append(point_a_x[-1])
                point_a_y.append(reflux_line(point_a_x[-1]))

            else:
                """If reflux_line(x) point is below the vapor quality line,
                then find the y value of the intercept of the reboil line"""
                point_a_x.append(point_a_x[-1])
                point_a_y.append(VB_line(point_a_x[-1]))

        step += 1
        if point_a_x[-1] < xB and not step % 2:
            done = True
    return plt.plot(point_a_x, point_a_y, "cx--")
```

Creating a function to print out desired values from any imput value

In [13]:
```
def print_values(alpha_ba, A, xB, xD, zF, q, R_R_min=1.0):
    """Statement is required to print z_F,EQ, R_min, VB/VB_min"""
    print("z_F,EQ: ", calc_zFEQ(alpha_ba, A, zF, q))
    print("Reflux Ratio Minimum: ", calc_Rmin(alpha_ba, A, xD, zF, q))
```

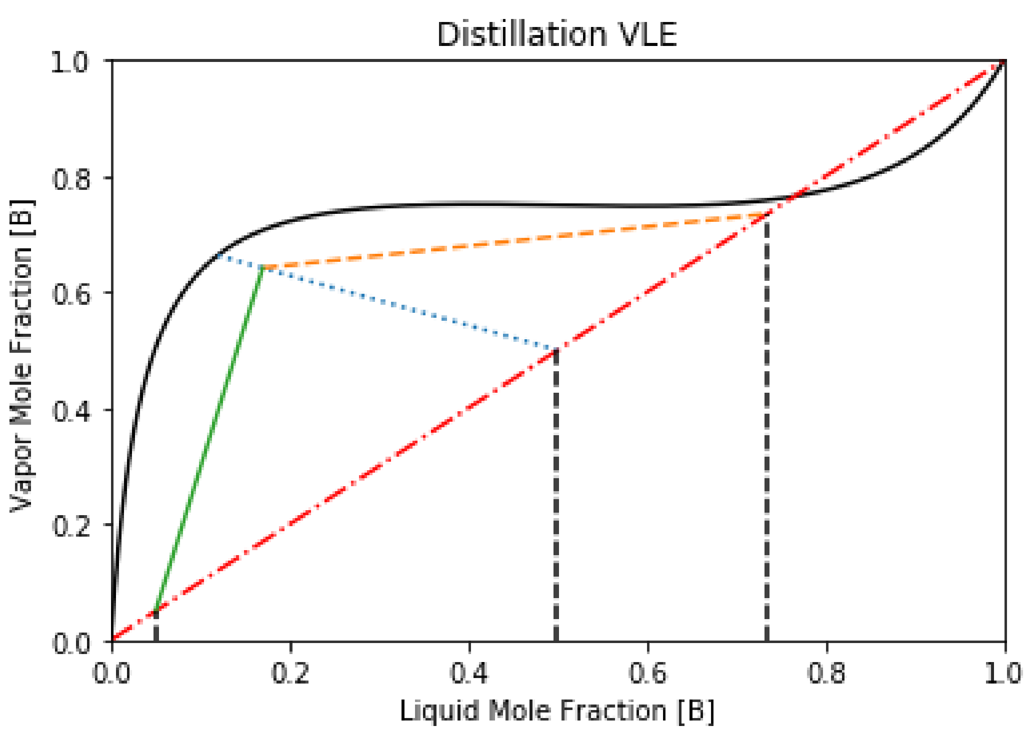
```python
print("VB minimum: ", calc_VB_min(alpha_ba, A, xB, zF, q))
print("VB to VB_min ratio", calc_VVminBminOp(alpha_ba, A, xB, xD, zF, q, R_R_min))
```

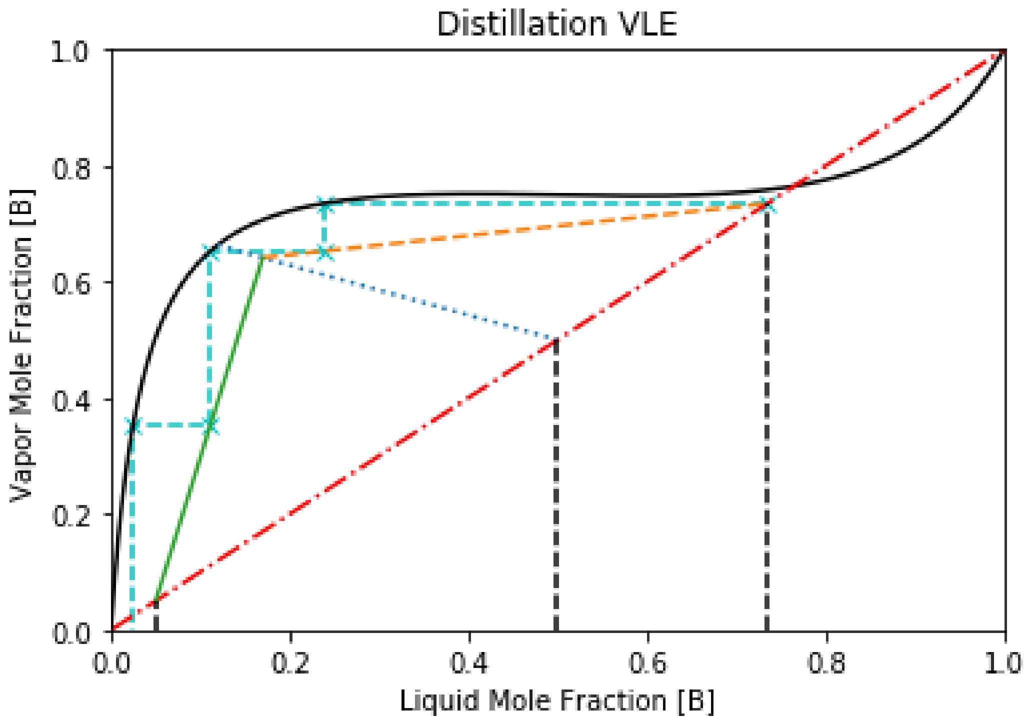
In [14]:
```python
def create_plot(alpha_ba, A, xB, xD, zF, q, R_R_min=1.0, indicate_trays=False):
    equilibrium_line = create_EQLine(alpha_ba, A)
    vapor_quality_line = create_qLine(zF, q)

    # This needs to be replaced with fsolve instead of bisection method
    # Finding intersection of vapor quality line and equilibrium line in terms of x c
    g = lambda x: equilibrium_line(x) - vapor_quality_line(x)
    intersect_eq_q = bisection(g)
    # Find the value of the reflux ratio from the minimum reflux ratio
    m = (xD - vapor_quality_line(intersect_eq_q)) / (xD - intersect_eq_q)
    R_min = m / (1 - m)
    reflux_ratio = R_min * R_R_min
    reflux_line = create_RLine(xD, reflux_ratio)

    # Root solving for the Reflux line and vapor quality line
    reflux_quality_subtraction = lambda x: reflux_line(x) - vapor_quality_line(x)
    intersect_q_R = bisection(reflux_quality_subtraction)

    # Calculate VB value
    VB_slope = (vapor_quality_line(intersect_q_R) - xB) / (intersect_q_R - xB)
    VB = 1 / (VB_slope - 1)

    reboil_line = create_VBLine(xB, VB)

    # Create distillation theoretical tray stair case
    if indicate_trays:
        create_steps(equilibrium_line, reboil_line, vapor_quality_line, reflux_line, x

    # Plot the desired functions
    x = np.linspace(0, 1, 1001)
    plt.plot(x, equilibrium_line(x), "k")
    plt.plot(x, x, "r-.")
    plt.plot([intersect_eq_q, zF], [vapor_quality_line(intersect_eq_q), vapor_quality_
    plt.plot([intersect_q_R, xD], [reflux_line(intersect_q_R), reflux_line(xD)],"--")
    plt.plot([xB, intersect_q_R], [xB, reboil_line(intersect_q_R)])

    # Plotting useful lines but not necessary
    plt.plot([zF, zF], [0, create_yxLine(zF)], "k--")  # Creating a dashed line for th
    plt.plot([xB, xB], [0, create_yxLine(xB)], "k--")
    plt.plot([xD, xD], [0, create_yxLine(xD)], "k--")

    plt.ylim(0, 1)
    plt.xlim(0, 1)
    plt.title("Distillation VLE")
    plt.xlabel("Liquid Mole Fraction [B]")
```

```
            plt.ylabel("Vapor Mole Fraction [B]")
            if not indicate_trays:
                plt.savefig("group7_capstone3", dpi=900, facecolor='w', edgecolor='w',
                        orientation='portrait', papertype="letter", format=None)
            else:
                plt.savefig("group7_capstone3_with_trays", dpi=900, facecolor='w', edgecolor=
                        orientation='portrait', papertype="letter", format=None)
            plt.show()
```

```
In [15]: alpha_BA = 3.0
         A = -0.9
         xB = 0.05
         xD = 0.735
         q = 0.3
         zF = 0.5
         R_R_min = 1.5

         print_values(alpha_BA, A, xB, xD, zF, q, R_R_min)
         create_plot(alpha_BA, A, xB, xD, zF, q, R_R_min)
         create_plot(alpha_BA, A, xB, xD, zF, q, R_R_min, indicate_trays=True)
```

```
z_F,EQ:  [0.11929854969739029, 0.6631577644154042]
Reflux Ratio Minimum:  0.13209711932865814
VB minimum:  0.12742001573572853
VB to VB_min ratio 1.9925910351242302
```





Distillation VLE
Vapor Mole Fraction [B]
Liquid Mole Fraction [B]
0.0
0.2
0.4
0.6
0.8
1.0


```
In [ ]:
```