SIS 10.1-10

AUTHOR
k.wodehouse

PUBLISHED
March 1, 2025

```
import numpy as np
from scipy.optimize import fsolve
import pandas as pd

df = pd.DataFrame({'component':['n-pentane', 'n-hexane', 'n-heptane'],
                   'zif':[0.55, 0.25, 0.20], 'pvap':[2.755, 1.021, 0.390]}).set_index('c
components = list(df.index)
df
```

	zif	pvap
component		
n-pentane	0.55	2.755
n-hexane	0.25	1.021
n-heptane	0.20	0.390

(a)

if the effluent of our flash is 50/50 vapor and liquid, that means

$$V = L = 0.5$$

and now we (i) may use this in our (my) calculations

the general outline of how this works

1. guess a pressure
2. calculate $\sum y_i$
3. if it equals 1, you're good and thats the pressure
4. if not, guess another pressure.

```
def isothermal_flash(Ptotal, V):
    y_sum = 0
    for component in components:
        ki = df.loc[component]['pvap'] / Ptotal
        zif = df.loc[component]['zif']
        y_sum += zif * ki / (1 + V*(ki - 1))

    return 1 - y_sum

pressure = fsolve(isothermal_flash, 2.0, args=(0.5))[0]
print(f'pressure: {pressure:.3f} bar')

def equlibrium_comp(Ptotal, V):
    report = pd.DataFrame()
    for component in components:
        ki = df.loc[component]['pvap'] / Ptotal
        zif = df.loc[component]['zif']
        yi = zif * ki / (1 + V*(ki - 1))
        xi = yi / ki

        report = pd.concat([report, pd.DataFrame({'component':component, 'x':[xi], 'y':[y


    return report.set_index('component')

equlibrium_comp(pressure, 0.5)
```

```
pressure: 1.491 bar
```

	x	y
component		
n-pentane	0.386206	0.713794
n-hexane	0.296745	0.203255
n-heptane	0.317049	0.082951

(b)

now instead of already knowing V and L we need to iterate for the V (or you could iterate for L) during each iteration of the pressure.

the general outline for this process is

1. guess a pressure (or let fsolve do it for you)
2. iterate to find what the V will be at that pressure such that $\sum x_i = 1$
3. calculate x_i with the guessed pressure and iteratively found V
4. if $\text{abs}(x_i - 0.3) = 0$, you're good
5. otherwise, guess another pressure (or let fsolve do it for you)

```
def get_v(V, Ptotal):
    sum_thingy = 0
    for component in components:
        z_iF = df.loc[component]['zif']
        pvap = df.loc[component]['pvap']
        ki = pvap / Ptotal
        sum_thingy += (z_iF * ki)/(1 + V*(ki - 1))
    return abs(1 - sum_thingy)

def pentane_flash(Ptotal):
    V = fsolve(get_v, 0.5, (Ptotal))[0] # hope there aren't 2 roots for some reason
    ki = df.loc['n-pentane']['pvap'] / Ptotal
    zif = df.loc['n-pentane']['zif']

    xi = zif / (1 + V*(ki - 1))
    return abs(0.3 - xi)

pressure = fsolve(pentane_flash, 2.0)[0]
V = fsolve(get_v, 0.5, (pressure))[0]
print(f'pressure: {pressure:.3f} bar')
print(f'liquid: {1 - V:.3f}')

def generate_report():
    print('\n----- vapor phase composition -----')
    for component in components:
        ki = df.loc[component]['pvap'] / pressure
        zif = df.loc[component]['zif']
        yi = zif * ki / (1 + V*(ki - 1))
        print(f'{component} y: {yi:.3f}')

    print('\n----- liquid phase composition -----')
    for component in components:
        ki = df.loc[component]['pvap'] / pressure
        zif = df.loc[component]['zif']
        yi = zif * ki / (1 + V*(ki - 1))
        xi = yi / ki
        print(f'{component} x: {xi:.3f}')

generate_report()
```

```
pressure: 1.285 bar
liquid: 0.272

----- vapor phase composition -----
n-pentane y: 0.643
n-hexane y: 0.234
n-heptane y: 0.123

----- liquid phase composition -----
n-pentane x: 0.300
n-hexane x: 0.294
n-heptane x: 0.406
```

```
# filler text
```