Summer Fellowship Report

submitted by
MehulKumar Sutariya

under the guidance of
Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

Acknowledgment

I wish to express my deepest gratitude to my internship guide Dr.Kannan M.Moudgalya, Professor, Department of Chemical Engineering, IIT Bombay for his continuous support and supervision throughout the internship. I have reaped benefits from his wisdom, guidance and patience all along.
I would also like to express my profound and heart felt thanks to Priyam Nayak, Pravin Dalve and Rahul Nagraj Department of Chemical Engineering, IIT Bombay and the FOSSEE team for their timely help and guidance

Contents

I Lee-Kesler-Plöcker EoS **4**

1 Lee-Kesler-Plöcker EoS **5**
- 1.1 About Lee-Kesler-Plöcker EoS . . . 5
- 1.2 Benedict-Webb-Rubin EoS modified by Lee-Kesler . . . 6
  - 1.2.1 Constant of BWR - Lee-Kesler . . . 7
- 1.3 Mixing rule suggested by Plöcker . . . 7
  - 1.3.1 Interaction Parameter $k'ij$ . . . 7
    - 1.3.1.1 Python Script for the data extraction . . . 8
- 1.4 Fugacity coefficient . . . 8
- 1.5 Enthalpy and Entropy departure function . . . 9
  - 1.5.1 Enthalpy . . . 9
  - 1.5.2 Entropy . . . 9
- 1.6 Isobaric heat capacity(C_p) and Isochoric heat capacity(C_v) departure . . . 9
  - 1.6.1 Isochoric heat capacity(C_v) . . . 9
  - 1.6.2 Isobaric heat capacity(C_p) . . . 9
- 1.7 Calculation Procedure . . . 10

2 Codes for Open Modelica **11**
- 2.1 Binary Interaction Parameter function (BIP_ LKP) . . . 11
- 2.2 Function for Reduced Volume Calculation . . . 13
- 2.3 Lee_Kesler_Plocker EoS model . . . 15

II Unifac Automation **24**

3 Data for the UNIFAC model **26**
- 3.1 ChemSep database . . . 26
- 3.2 PYTHON script . . . 27
- 3.3 Subgroup and R, Q and Aij-Aji value . . . 27
- 3.4 PYTHON Script . . . 28

4 Flow of Model **31**
- 4.1 Code . . . 33

III Bug Fixing in 1.13.2 **47**

5 ShortCut Column **48**
5.1 Convergence problem . . . 48
5.2 Possible error inside the ShortCut column . . . 48
5.2.1 Confirmation of this error . . . 48
5.2.2 Solution of that error . . . 48
5.3 minR<=0 error . . . 49
5.3.1 Error due to theta . . . 49
5.3.2 error in distillate vapor fraction and Reflux ratio . . . 51
5.3.3 Improvement in solution . . . 51

Part I

Lee-Kesler-Plöcker EoS

Chapter 1

Lee-Kesler-Plöcker EoS

1.1 About Lee-Kesler-Plöcker EoS

Lee-Kesler (LK)[1] Equation of State (EoS) is the most accurate enthalpy model for gases. Lee-Kesler introduce the BWR Eos with some modification to improve the volumetric and thermodynamic correlation developed Pitzer and co-worker by the and extend it for the wide range of reduced temperature and pressure. This method is reliable for non-polar as well as slightly polar substances and accurate for subcooled and superheated region.

Pitzer and co-worker's work suggest that their correlation can be represented by

$$Z = Z_0 + \omega Z_1$$

But Lee-Kesler's analysis suggest that it can better represent by using the reference fluid as shown below

$$Z = Z_s + \frac{\omega_s}{\omega_r}(Z_r - Zs)$$

Here;

Z = Compressibility factor;
ω = Acentric factor;
s = Simple fluid;
r = Reference fluid;

Here reference fluid is taken as N-Octane because it is the heaviest hydrocarbon for with P-V-T data are available.
The mixing rule provided by the Lee-Keseler are modified by the Plöcker[2] for better representation and accuracy of the Lee-Kesler's EoS.

Equation in Lee-Kesler-Plöcker EoS

1.2 Benedict-Webb-Rubin EoS modified by Lee-Kesler

$$z^{(i)} = \frac{P_r V_r}{T_r} = 1 + \frac{B}{V_r} + \frac{C}{V_r^2} + \frac{D}{V_r^5} + \frac{c_4}{V_r^2 T_r^3}\left(\beta + \frac{\gamma}{V_r^2}\right)\left(exp\left(\frac{-\gamma}{V_r^2}\right)\right)$$

$$P_r = \frac{P}{P_c}$$

$$T_r = \frac{T}{T_c}$$

$$V_r = \frac{V}{V_c} = \frac{P_c V}{R T_c}$$

$$B = b_1 - \frac{b_2}{T_r} - \frac{b_3}{T_r^3} - \frac{b_4}{T_r^4}$$

$$C = c_1 - \frac{c_2}{T_r} + \frac{c_3}{T_r^3}$$

$$D = d_1 + \frac{d_2}{T_r}$$

1.2.1 Constant of BWR - Lee-Kesler

Table 1.1: Constants for the BWR-Lee-Kesler Equation

Constant	Simple fluids	Reference fluids
b1	0.1181193	0.02026579
b2	0.265728	0.331511
b3	0.154790	0.027655
b4	0.030325	0.203488
c1	0.0236744	0.0313385
c2	0.0186984	0.0503618
c3	0.0	0.016901
c4	0.042724	0.041577
d1	0.0000155488	0.000048736
d2	0.0000623689	0.00000740336
β	0.65392	1.226
γ	0.060167	0.03754

1.3 Mixing rule suggested by Plöcker

$$V_{cjk} = \frac{1}{8}(V_{cj}^{\frac{1}{3}} + V_{ck}^{\frac{1}{3}})^3$$

$$V_{cm} = \sum_j \sum_k z_j z_k \cdot V_{cjk}$$

$$T_{cjk} = (T_{cj} \cdot T_{ck})^{\frac{1}{2}} k'_{jk}$$

$$T_{cm} = \frac{1}{{V_{cm}}^{\eta}} \cdot \sum_j \sum_k z_j z_k \cdot {V_{cjk}}^{\eta} \cdot T_{cjk}$$

$$P_{cm} = (0.2905 - 0.085\omega_m) \cdot R \cdot \frac{T_{cm}}{V_{cm}}$$

$$\omega_m = \sum_j z_j \omega_j$$

Here the value of η is 0.25; It is empirically found by the Plöcker and kannap.[2]

1.3.1 Interaction Parameter $k'ij$

Binary interaction parameter for some compounds are given in the work of Plöcker and kannap.[2] Same data are digitally transformed in file named "lkp_ip.dat" by the developer of the DWSIM. For this work that '.dat' file are converted into '.csv' file and by using the python script, data is extracted and used in BIP_LKP function of Open Modelica.
Python script for that are given here

1.3.1.1 Python Script for the data extraction

```python
import pandas as pd

file = pd.read_csv('lkpip.csv')

comp1 = file['1']
comp2 = file['2']

BIP = file['BIP']

comp = [[0 for _ in range(2)] for _ in range(len(comp1))]
comp_comp = [0 for _ in range(len(comp1))]
bij = [0 for _ in range(len(comp1))]

for i in range(len(comp1)):
    comp[i][0] = str(comp1[i])
    comp[i][1] = str(comp2[i])
    comp_comp[i] =  '"'+str(comp1[i])+'_'+str(comp2[i])+'"'
    bij[i] = float(BIP[i])

f = open('LKP_BIP.txt', '+w')
f.write(str(comp))
f.write('\n')
f.write('\n')
f.write('\n')
f.write(str(bij))
f.write('\n')
f.write('\n')
f.write('\n')
f.write(str(comp_comp))
f.close()
```

1.4 Fugacity coefficient

$$ln(\phi_m) = z - 1 - ln(z) + \frac{B}{V_r} + \frac{C}{2V_r^2} + \frac{D}{5V_r^5} + \frac{c_4}{2T_r^3\gamma}\left\{\beta + 1 - \left(\beta + 1 + \frac{\gamma}{v_r^2}\right)exp\left(-\frac{\gamma}{V_r^2}\right)\right\}$$

$$\begin{aligned}ln(\phi_i) = ln(\phi_m) - \frac{1}{T}\frac{\triangle H}{RT_{cM}}\sum_{j\neq i} x_j\left(\frac{dT_{cM}}{dx_j}\right)_{x_k} + \frac{Z_m - 1}{P_{cM}}\sum_{j\neq i} x_j\left(\frac{dP_{cM}}{dx_j}\right)_{x_k} \\ - \left(\frac{\partial ln(\phi_m)}{\partial \omega_M}\right)_{T_r,P_r}\sum_{j\neq i} x_j\left(\frac{d\omega_M}{dx_i}\right)_{x_k}\end{aligned}$$

$$\left(\frac{\partial ln(\phi_m)}{\partial \omega_M}\right)_{T_r,P_r} = \frac{1}{\omega_r}[(ln(\phi_m) - ln(\phi_r)]$$

$$\left(\frac{dT_{cM}}{dx_j}\right)_{x_k} = \left[2\sum_l (v_{clj}^{\eta}T_{clj} - v + cli^{\eta}T_{cli}) - \eta v_{cm}^{\eta-1}\left(\frac{dv_{cm}}{dx_j}\right)_{x_k}T_{cM}\right]\Big/ v_{cm}^{\eta}$$

$$\left(\frac{dv_{cM}}{dx_j}\right)_{x_k} = 2\sum_l x_l(v_{clj} - v_{cli})$$

$$\left(\frac{dP_{cM}}{dx_j}\right)_{x_k} = P_{cM}\left[\left(\frac{dZ_{cM}}{dx_j}\right)_{x_k}\Big/ Z_{cM} + \left(\frac{dT_{cM}}{dx_j}\right)_{x_k}\Big/ T_{cM} - \left(\frac{dv_{cM}}{dx_j}\right)_{x_k}\Big/ v_{cM}\right]$$

$$\left(\frac{dZ_{cM}}{dx_j}\right)_{x_k} = -0.085\left(\frac{d\omega_M}{dx_i}\right)_{x_k}$$

$$\left(\frac{d\omega_M}{dx_i}\right)_{x_k} = \omega_j - \omega_i$$

1.5 Enthalpy and Entropy departure function

$$E = \frac{c_4}{2T_r^3\gamma}\left[\beta + 1 - \left(\beta + 1 + \frac{\gamma}{V_r^2}\right)exp\left(\frac{-\gamma}{V_r^2}\right)\right]$$

1.5.1 Enthalpy

$$\frac{\Delta h}{RT_c} = \frac{h - h^0}{RT_c} = T_r\left[Z - 1 - \frac{b_2 + \frac{2b_3}{T_r} + \frac{3b_4}{T_r^2}}{T_rV_r} - \frac{c_2 - \frac{2c_3}{T_r^2}}{2T_rV_r^2} + \frac{d_2}{5T_rV_r^5} + 3E\right]$$

1.5.2 Entropy

$$\frac{\Delta s}{R} + ln\left(\frac{P}{P^0}\right) = \frac{s - s^0}{R}ln\left(\frac{P}{P^0}\right) = ln(z) - \frac{b_1 + \frac{b_3}{T_r^2} + \frac{2b_4}{T_r^3}}{V_r} - \frac{c_1 - \frac{2c_3}{T_r^3}}{2V_r^2} - \frac{d_1}{5V_r^5} + 2E$$

1.6 Isobaric heat capacity(C_p) and Isochoric heat capacity(C_v) departure

1.6.1 Isochoric heat capacity(C_v)

$$\frac{\Delta C_v}{R} = \frac{2\left(b_3 + \frac{3b_4}{T_r}\right)}{T_r^2V_r} - \frac{3c_3}{T_r^3V_r^2} - 6E$$

1.6.2 Isobaric heat capacity(C_p)

$$\frac{\Delta C_p}{R} = \frac{\Delta C_v}{R} - 1 - T_r\left(\frac{\partial P_r}{\partial T_r}\right)_{V_r}^2\Big/\left(\frac{\partial P_r}{\partial V_r}\right)_{T_r}$$

$$\left(\frac{\partial P_r}{\partial T_r}\right)_{V_r} = \frac{1}{V_r}\left\{1 + \frac{b_1 + \frac{b_3}{T_r^2} + \frac{2b_4}{T_r^3}}{V_r} + \frac{c_1 - \frac{2c_3}{T_r^3}}{2V_r^2} + \frac{d_1}{5V_r^5} - \frac{2c_4}{T_r^3V_r^2}\left[\left(\beta + \frac{\gamma}{V_r^2}\right)exp\left(-\frac{\gamma}{V_r^2}\right)\right]\right\}$$

$$\left(\frac{\partial P_r}{\partial V_r}\right)_{T_r} = -\frac{T_r}{V_r^2}\left\{1 + \frac{2B}{V_r} + \frac{3C}{V_r^2} + \frac{6D}{V_r^5} + \frac{c_4}{T_r^3V_r^2}\left[3\beta + \left\{5 - 2\left(\beta + \frac{\gamma}{V_r^2}\right)\right\}\frac{\gamma}{V_r^2}\right]exp\left(-\frac{\gamma}{V_r^2}\right)\right\}$$

1.7 Calculation Procedure

1 Calculate critical property by Mixing rule as given in Section 1.3. Find the reduced temperature and pressure.

2 By using reduced property and constant given in 1.2.1 calculate

   a Vr and Z as shown section 1.2

   b ϕ_m as shown in section 1.4

   c $\triangle H, and\ \triangle S$ as shown in section 1.5

   d $\triangle C_p, and\ \triangle C_v$ as shown in section 1.6

   for both, simple fluid and reference fluid.

3 If property shown above is denoted by the P_r for reference fluid and P_s for simple fluid then

$$P_m = P_s + \frac{\omega_m}{\omega_r}(P_r - P_s)$$

Here;

P_m = Property of the mixture
ω_m = ω calculated from the mixing rule
ω_r = Accentric factor of reference fluid (Here, it is n-Octane, and value is 0.3978)

4 ϕ_i for both phase and for each component as shown in Section 1.4

5 Find the K value from $K = \frac{Liquid\ Fugacity\ Coefficient_i}{Vapor\ Fugacity\ Coefficient_i}$

Chapter 2

Codes for Open Modelica

2.1 Binary Interaction Parameter function (BIP_ LKP)

```
function BIP_LKP
input String comp[2];
output Real kij;
protected
parameter String comp_comp[142] = {"Methane_Ethane", "Methane_Ethylene",
    "Methane_Propane", "Methane_Propylene", "Methane_Nbutane", "
    Methane_Isobutane", "Methane_Npentane", "Methane_Isopentane", "
    Methane_Nhexane", "Methane_Cyclohexane", "Methane_Benzene", "
    Methane_Nheptane", "Methane_Noctane", "Methane_Nnonane", "
    Methane_Ndecane", "Ethane_Ethylene", "Ethane_Propane", "
    Ethane_Propylene", "Ethane_Nbutane", "Ethane_Isobutane", "
    Ethane_Npentane", "Ethane_Isopentane", "Ethane_Nhexane", "
    Ethane_Cyclohexane", "Ethane_Benzene", "Ethane_Nheptane", "
    Ethane_Noctane", "Ethane_Nnonane", "Ethane_Ndecane", "
    Ethylene_Nbutane", "Ethylene_Benzene", "Ethylene_Nheptane", "
    Acetylene_Ethylene", "Propane_Propylene", "Propane_Nbutane", "
    Propane_Isobutane", "Propane_Npentane", "Propane_Isopentane", "
    Propane_Nhexane", "Propane_Cyclohexane", "Propane_Benzene", "
    Propane_Nheptane", "Propane_Noctane", "Propane_Nnonane", "
    Propane_Ndecane", "Propylene_Nbutane", "Propylene_Isobutane", "
    Propylene_Isobutene","Nbutane_Isobutane", "Nbutane_Npentane",
"Nbutane_Isopentane", "Nbutane_Nhexane", "Nbutane_Cyclohexane", "
    Nbutane_Benzene", "Nbutane_Nheptane", "Nbutane_Noctane", "
    Nbutane_Nnonane", "Nbutane_Ndecane", "Npentane_Isopentane", "
    Npentane_Nhexane", "Npentane_Cyclohexane", "Npentane_Benzene", "
    Npentane_Nheptane", "Npentane_Noctane", "Npentane_Nnonane", "
    Npentane_Ndecane", "Nhexane_Cyclohexane", "Nhexane_Benzene", "
    Nhexane_Nheptane", "Nhexane_Noctane", "Nhexane_Nnonane", "
    Nhexane_Ndecane", "Benzene_Cyclohexane", "Benzene_Nheptane", "
    Benzene_Noctane", "Benzene_Isooctane", "Benzene_Nnonane", "
    Benzene_Ndecane", "Cyclohexane_Nheptane", "Cyclohexane_Noctane", "
    Cyclohexane_Nnonane", "Cyclohexane_Ndecane", "Nheptane_Noctane", "
    Nheptane_Isooctane", "Nheptane_Nnonane", "Nheptane_Ndecane", "
    Noctane_Nnonane", "Noctane_Ndecane", "Noctane_Ndecane", "
    Nitrogen_Nitrogen", "Nitrogen_Ethylene", "Nitrogen_Ethane", "
    Nitrogen_Propane", "Nitrogen_Propylene", "Nitrogen_Nbutane", "
    Nitrogen_Npentane", "Nitrogen_Nhexane", "Nitrogen_Oxygen", "
    Nitrogen_Carbonmonoxide", "Nitrogen_Argon", "Nitrogen_Hydrogensulfide
    ", "Nitrogen_Carbondioxide", "Nitrogen_Nitrousoxide", "
    Nitrogen_Ammonia", "Carbondioxide_Methane", "Carbondioxide_Ethane", "
    Carbondioxide_Propane", "Carbondioxide_Nbutane", "
    Carbondioxide_Isobutane", "Carbondioxide_Npentane", "
    Carbondioxide_Nhexane", "Carbondioxide_Cyclohexane", "
    Carbondioxide_Benzene", "Carbondioxide_Nheptane", "
    Carbondioxide_Noctane", "Carbondioxide_Nnonane", "
    Carbondioxide_Ndecane", "Carbondioxide_Hydrogensulfide", "
    Carbondioxide_R12", "Carbondioxide_Methanol", "Hydrogen_Methane", "
    Hydrogen_Methane", "Hydrogen_Ethylene", "Hydrogen_Propane", "
    Hydrogen_itbutane", "Hydrogen_Npentane", "Hydrogen_Nhexane", "
    Hydrogen_Nheptane", "Hydrogen_Nitrogen", "Hydrogen_Carbonmonoxide", "
    Hydrogen_Carbondioxide", "Argon_Oxygen", "Argon_Ammonia", "
    Argon_Methane", "Oxygen_Nitrousoxide", "Carbonmonoxide_Methane", "
    Krypton_Oxygen", "Hydrogensulfide_Isobutane", "Nitrousoxide_Methane",
     "Water_Carbondioxide", "Water_Ammonia", "Water_Methanol"};

parameter Real BIP[142] = {1.052, 1.014, 1.113, 1.089, 1.171, 1.155,
    1.2401, 1.228, 1.304, 1.2690000000000001, 1.234, 1.367, 1.423, 1.484,
     1.5330000000000001, 0.991, 1.01, 1.002, 1.0290000000000001, 1.036,
    1.064, 1.07, 1.1059999999999999, 1.081, 1.0659999999999998, 1.143,
    1.165, 1.214, 1.237, 0.998, 1.094, 1.163, 0.948, 0.992, 1.003, 1.003,
     1.008, 1.0090000000000001, 1.047, 1.037, 1.011, 1.067, 1.09, 1.115,
    1.139, 1.01, 1.0090000000000001, 1.006, 1.001, 0.9940000000000001,
    0.998, 1.018, 1.008, 0.9990000000000001, 1.0270000000000001, 1.046,
    1.064, 1.078, 0.987, 0.998, 0.996, 0.977, 1.004, 1.02, 1.033, 1.045,
    0.998, 0.978, 1.008, 1.005, 1.015, 1.025, 0.9790000000000001, 0.985,
    0.987, 0.982, 1.034, 1.047, 0.9990000000000001, 1.01, 1.021, 1.032,
    0.993, 1.002, 1.002, 1.01, 0.993, 0.9990000000000001, 0.991, 0.977,
    1.032, 1.082, 1.177, 1.151, 1.276, 1.3719999999999999, 1.442, 0.997,
    0.987, 0.988, 0.983, 1.11, 1.073, 1.033, 0.975, 0.938, 0.925, 0.955,
    0.946, 1.002, 1.018, 1.054, 1018.0, 1.058, 1.09, 1.126, 1.16, 0.922,
    0.9690000000000001, 1.069, 1.216, 1.604, 1408.0, 1828.0, 2093.0,
    2.335, 2.4659999999999997, 2.8339999999999996, 1.08, 1.085, 1.624,
    0.985, 1.01, 0.9840000000000001, 1.057, 0.9740000000000001,
    0.9890000000000001, 0.9470000000000001, 1.0170000000000001, 0.92,
    1.1520000000000001, 0.9790000000000001};

  String name;
  String nameRev;
algorithm
  name := comp[1] + "_" + comp[2];
  nameRev := comp[2] + "_" + comp[1];
  if Simulator.Files.Thermodynamic_Functions.FindString(comp_comp,name)
      <> (-1) then
    kij := BIP[Simulator.Files.Thermodynamic_Functions.index(
        comp_comp,name)];
  else
    kij:=1;
  end if;
end BIP_LKP;
```

2.2 Function for Reduced Volume Calculation

```
function LKP_V
//algorithm for this function is devloped with the help of DWSIM's
    algorithm
  input Real Pr;
  input Real Tr;
  input Real B;
  input Real C;
  input Real D;
  input Real c4;
  input Real bta;
  input Real gma;
  input String phas;
  output Real lll;

  protected
  Real Tinf, Tsup, Nsub, delta_T, Vg, Vl;
  Real fT, fT_inf, i;
  Real aaa, bbb, ccc, ddd, eee, min11, min22, faa, fbb, fcc, ppp, qqq,
      rrr, sss, tol11, xmm, tvar2;
  parameter Integer ITMAX2 = 100;
  Integer iter2;

algorithm

if phas == "Liquid" then
  Tinf := 0.0;
  Tsup := 10.0;
  Nsub := 500.0;
elseif phas == "Gas" then
  Tinf := 1001.0;
  Tsup := 0;
  Nsub := 500;
end if;

  delta_T := (Tsup-Tinf)/Nsub;

  i:=0;
  fT:=1;
  fT_inf :=1;  Tinf := 1001;
  while (fT.*fT_inf)>0 or i>500 loop
    i := i + 1;
    Vl := Tinf;
    if Vl==0 then
      fT:=-1;
    else
      fT := (Pr*Vl./Tr) - (1+ B./Vl + C./(Vl^2) + D./(Vl^5) + (c4./(Tr^2
          * Vl^2))*(bta + gma./(Vl^2))*exp(-gma./Vl^2));
    end if;
    Tinf := Tinf + delta_T;
    Vl := Tinf;
    fT_inf := (Pr*Vl./Tr) - (1+ B./Vl + C./(Vl^2) + D./(Vl^5) + (c4./(Tr
        ^2 * Vl^2))*(bta + gma./(Vl^2))*exp(-gma./Vl^2));

    if fT_inf*fT < 0 then
```

```
        break;
      end if;
    end while;

    Tsup := Tinf;
    Tinf := Tinf - delta_T;

    aaa := Tinf;
    bbb := Tsup;
    ccc := Tsup;

    faa := (Pr * aaa / Tr) - (1 + B ./ aaa + C ./ aaa ^ 2 + D ./ aaa ^ 5 +
        c4 ./ Tr ^ 3 ./ aaa ^ 2 * (bta + gma ./ aaa ^ 2) * exp(- gma ./ aaa
        ^ 2));
    fbb := (Pr * bbb / Tr) - (1 + B / bbb + C / bbb ^ 2 + D / bbb ^ 5 + c4
        / Tr ^ 3 / bbb ^ 2 * (bta + gma / bbb ^ 2) * exp(-gma ./ bbb ^ 2));
    fcc := fbb;
    iter2 := 0;

    while iter2 < ITMAX2 loop
      if (fbb > 0 and fcc > 0) or (fbb < 0 and fcc < 0) then
        ccc := aaa;
        fcc := faa;
        ddd := bbb - aaa;
        eee := ddd;
      end if;
      if abs(fcc) < abs(fbb) then
        aaa := bbb;
        bbb := ccc;
        ccc := aaa;
        faa := fbb;
        fbb := fcc;
        fcc := faa;
      end if;

    tol11 := 0.0000001;
    xmm := 0.5 * (ccc - bbb);

    if (abs(xmm) <= tol11) or (fbb == 0) then
      break;
    end if;

    if (abs(eee) >= tol11) and (abs(faa) > abs(fbb)) then
        sss := fbb / faa;
        if aaa == ccc then
          ppp := 2 * xmm * sss;
          qqq := 1 - sss;
        else
          qqq := faa ./ fcc;
          rrr := fbb ./ fcc;
          ppp := sss * (2 * xmm * qqq * (qqq - rrr) - (bbb - aaa) * (rrr -
              1));
          qqq := (qqq - 1) * (rrr - 1) * (sss - 1);
        end if;

        if ppp > 0 then
          qqq := (-qqq);
        end if;

        ppp := abs(ppp);
        min11 := 3 * xmm * qqq - abs(tol11 * qqq);
        min22 := abs(eee * qqq);

        if min11 < min22 then
          tvar2 := min11;
        end if;

        if min11 > min22 then
          tvar2 := min22;
        end if;

        if 2 * ppp < tvar2 then
          eee := ddd;
          ddd := ppp ./ qqq;
        else
          ddd := xmm;
          eee := ddd;
        end if;
    else
        ddd := xmm;
        eee := ddd;
    end if;
    aaa := bbb;
    faa := fbb;

    if (abs(ddd) > tol11) then
      bbb := bbb + ddd;
    else
      if xmm>0 then
        bbb := bbb + tol11;
      elseif xmm<0 then
        bbb := bbb - tol11;
      else
        bbb := bbb;
      end if;
      //bbb := bbb + Math.Sign(xmm) * tol11;
    end if;

    fbb := Pr * bbb ./ Tr - (1 + B ./ bbb + C ./ bbb ^ 2 + D ./ bbb ^ 5 +
        c4 ./ Tr ^ 3 ./ bbb ^ 2 * (bta + gma ./ bbb ^ 2) * exp(-gma ./ bbb
        ^ 2));

    iter2 := iter2 + 1;
    end while;

    lll := bbb;
  end LKP_V;
```

2.3 Lee_Kesler_Plocker EoS model

```
model LEE_KESLER_PLOCKER
  import data = Simulator.Files.Chemsep_Database;
  parameter Integer NOC = 2;
  parameter data.Water wat;
  parameter data.Methanol meth;
  parameter data.General_Properties comp[NOC] = {wat,meth};


//required data
  Real T=348.5;
  Real P=101325;
  Real compMolFrac[3,NOC] =
      {{0.5,0.5},{0.72169291,0.27830709},{0.281016,0.718984}};
  Real Psat[NOC];
  Real V(start = 10), Vl(start=0.5);

//property variable for simple fluid
//reduced property
  Real Pcr(start = 1);
  Real Vcr(start = 1);
  Real Tcr(start = 1);

  Real Pcrl(start = 1);
  Real Vcrl(start = 1);
  Real Tcrl(start = 1);

//Lkp Eos' varibles
  Real Z(start = 2, min=0), Zl(start = 2, min=0);
  Real B, Bl;
  Real C, Cl;
  Real D, Dl;
  Real E, El;
  Real liqFugCoff(start = 2);
  Real vapFugCoff(start = 2);
// mixing rule variable
  parameter Real R=8.314 "J/k/mol"; //Universal gas constant

  Real AFM(start=1);
  Real TcM(start=298);
  Real PcM(start=101325);
  Real VcM(start=0.5, min=0);

  Real AFMl(start=1);
  Real TcMl(start=298);
  Real PcMl(start=101325);
  Real VcMl(start=0.5, min=0);

  parameter Real ita = 0.25;

//Variable for Referance fluid "Noctane"
   parameter Real Tc_ref = 568.7;
   parameter Real Pc_ref = 2490000;
   parameter Real Vc_ref = 0.492;
   parameter Real omega_ref = 0.3978;

//   Real Pr_ref(start = 1); // we have to use mixture'd Pr Tr
//   Real Tr_ref(start = 1);
   Real Vr_ref(start = 10);
   Real Vrl_ref(start = 1);
```

```

    Real B_ref, Bl_ref;
    Real C_ref, Cl_ref;
    Real D_ref, Dl_ref;
    Real E_ref;
    Real El_ref;

    Real Z_ref(start = 1);
    Real Zl_ref(start = 1);
    Real liqFugCoff_ref(start = 1); // value is in logarithmic
    Real vapFugCoff_ref(start = 1);

    Real h, hl;
    Real h_ref, hl_ref;
    Real H, Hl;

    Real s,sl;
    Real s_ref, sl_ref;
    Real S, Sl;

    Real resMolSpHeat[3];
    Real resMolEnth[3];
    Real resMolEntr[3];
//LKP constant
//1=simple fluid, 2=referance fluid
//b1, b2, b3, b4, c1, c2, c3, c4, d1, d2, beta, gamma
  parameter Real b1[2] = {0.1181193, 0.2026579};
  parameter Real b2[2] = {0.265728, 0.331511};
  parameter Real b3[2] = {0.154790, 0.027655};
  parameter Real b4[2] = {0.030323, 0.203488};
  parameter Real c1[2] = {0.0236744, 0.0313385};
  parameter Real c2[2] = {0.0186984, 0.0503618};
  parameter Real c3[2] = {0.0, 0.016901};
  parameter Real c4[2] = {0.042724, 0.041577};
  parameter Real d1[2] = {0.0000155488, 0.000048736};
  parameter Real d2[2] = {0.0000623689, 0.00000740336};
  parameter Real bta[2] = {0.65392, 1.226};
  parameter Real gma[2] = {0.060167, 0.03754};

// final equation variable
  Real Zf(start = 2), Zfl(start = 2);
  Real vapFugCofff(start=1), liqFugCofff(start=1);
  Real compVapFugCoff[NOC];
  Real compLiqFugCoff[NOC];
  Real K[NOC];

//extra variable
  Real Tcij[NOC, NOC];
  Real Kij[NOC,NOC];
  Real Vcij[NOC, NOC];
  Real dT[NOC,NOC], dP[NOC,NOC], dV[NOC,NOC], dZ[NOC, NOC], sum1[NOC,NOC
      ],sum2[NOC,NOC];
  Real suma[NOC], sumb[NOC], sumc[NOC];
  Real dTl[NOC,NOC], dPl[NOC,NOC], dVl[NOC,NOC], dZl[NOC, NOC], sum1l[
      NOC,NOC],sum2l[NOC,NOC];
  Real sumal[NOC], sumbl[NOC], sumcl[NOC];
  Real gammaDew[NOC], gammaBubl[NOC],liqfugcoeff_bubl[NOC],
      vapfugcoeff_dew[NOC],gamma[NOC];
equation
resMolSpHeat[1] = 0;
  resMolSpHeat[2] = 0;
  resMolSpHeat[3] = 0;

resMolEnth[1] = 0;
  resMolEnth[2] = 0;
  resMolEnth[3] = 0;

resMolEntr[1] = 0;
  resMolEntr[2] = 0;
  resMolEntr[3] = 0;

  for i in 1:NOC loop
    Psat[i] = Simulator.Files.Thermodynamic_Functions.Psat(comp[i].VP,T);
    gammaDew[i] = 1;
    gammaBubl[i] = 1;
    liqfugcoeff_bubl[i] = 1;
    vapfugcoeff_dew[i] = 1;
    gamma[i] = 1;
  end for;
////////////////////////////////////// Gas phase
    //////////////////////////////////////////////////
/*
    /=========================================================================================
                                                SIMPLE FLUID
=========================================================================================
    */
//mixing rules

  AFM = sum(compMolFrac[3,:].* comp[:].AF);

  VcM = sum({sum({compMolFrac[3,i].*compMolFrac[3,j].*(1/8)*(1/1000)*((comp[i].Vc)^(1/3)+(comp[j].Vc)^(1/3))^3 for j in 1:NOC})for i in 1:NOC});

  TcM = (1/(VcM^ita))*sum({sum({compMolFrac[3,i].*compMolFrac[3,j].*(((1/8)*(((comp[i].Vc)^(1/3)+(comp[j].Vc)^(1/3))^3)*(1/1000))^(ita))*((comp[i].Tc*comp[j].Tc)^(1/2))*BIP_LKP({comp[i].name,comp[j].name}) for j in 1:NOC}) for i in 1:NOC});

  PcM = (0.2905-0.085*AFM)*R*TcM./VcM;

//only for cheking purpose
  for i in 1:NOC loop
    for j in 1:NOC loop
      Kij[i,j] = BIP_LKP({comp[i].name,comp[j].name});
    end for;
  end for;

  for i in 1:NOC loop
    for j in 1:NOC loop
      Tcij[i,j] = ((comp[i].Tc*comp[j].Tc)^(1/2))*BIP_LKP({comp[i].name,comp[j].name});
    end for;
  end for;

  for i in 1:NOC loop
    for j in 1:NOC loop
      Vcij[i,j] = (1/8)*(((comp[i].Vc)^(1/3)+(comp[j].Vc)^(1/3))^3)
          *(1/1000);
    end for;
  end for;
//cheking perameter done


//reduceded mixer property
  Pcr = P/PcM;
  Vcr = V/VcM;
  Tcr = T/TcM;

//Eos Equation constant
  B = b1[1] - b2[1]./Tcr - b3[1]./(Tcr^2) - b4[1]./(Tcr^3);
  C = c1[1] - c2[1]./Tcr + c3[1]./(Tcr^3);
  D = d1[1] + d2[1]./Tcr;

//for volume
  Vcr = LKP_V(Pcr, Tcr, B, C, D, c4[1], bta[1], gma[1], "Gas");
//for compressibility factor
  Z = (Pcr.*Vcr)./(Tcr);
//for fugacity coefficent
  E = c4[1]/(2*Tcr^3*gma[1])*(bta[1] +1 - (bta[1] + 1 + gma[1]/Vcr^2)*exp
      (-gma[1]/Vcr^2));

  vapFugCoff = Z - 1 - log(Z) + B/Vcr + C/(2*Vcr^2) + D/(5*Vcr^5) + E;

//enthalpy function
  h = Tcr*(Z - 1 - (b2[1] + 2*b3[1]./Tcr + 3*b4[1]./Tcr^2)./(Tcr*Vcr) - (
      c2[1]-3*c3[1]./Tcr^2)/(2*Tcr*Vcr^2) + d2[1]./(5*Tcr*Vcr^5) + 3*E);
//entropy function
  s = log(Z) - log(1) - (b1[1] + b3[1]./Tcr^2 + 2*b4[1]./Tcr^3)./Vcr -(c1
      [1]-2*c3[1]./Tcr^3)./(2* Vcr^2) - d1[1]/(5*Vcr^5) + 2*E;

/*
    /==========================================================================

                                    REFERANCE FLUID Gas
============================================================================/
    */
/*/reduceded property   not require for the referance
  Pr_ref = P./Pc_ref;
  Tr_ref = T./Tc_ref;/*/

//Eos Equation constant
  B_ref = b1[2] - b2[2]./Tcr - b3[2]./(Tcr^2) - b4[2]./(Tcr^3);
  C_ref = c1[2] - c2[2]./Tcr + c3[2]./(Tcr^3);
  D_ref = d1[2] + d2[2]./Tcr;

//for volume
  Vr_ref = LKP_V(Pcr, Tcr, B_ref, C_ref, D_ref, c4[2], bta[2], gma[2], "
      Gas");
//compressibility factor
```

```
  Z_ref = (Pcr.*Vr_ref)./(Tcr);
//fugacity coefficient
  E_ref = c4[2]/(2*Tcr^3*gma[2])*(bta[2] +1 - (bta[2] + 1 + gma[2]/Vr_ref
      ^2)*exp(-gma[2]/Vr_ref^2));

  vapFugCoff_ref = Z_ref - 1 - log(Z_ref) + B_ref/Vr_ref + C_ref/(2*
      Vr_ref^2) + D_ref/(5*Vr_ref^5) + E_ref;

// enthalpy
  h_ref = Tcr*(Z_ref - 1 - (b2[2] + 2*b3[2]./Tcr + 3*b4[2]./Tcr^2)./(Tcr*
      Vr_ref) - (c2[2]-3*c3[2]./Tcr^2)/(2*Tcr*Vr_ref^2) + d2[2]./(5*Tcr*
      Vr_ref^5) + 3*E_ref);
//entropy
    s_ref = log(Z_ref) - log(1) - (b1[2] + b3[2]./Tcr^2 + 2*b4[2]./Tcr^3)
        ./Vr_ref -(c1[2]-2*c3[2]./Tcr^3)./(2* Vr_ref^2) - d1[2]/(5*Vr_ref
        ^5) + 2*E_ref;

//final equation of gas
  Zf = Z + (AFM./omega_ref)*(Z_ref - Z);
  vapFugCofff = vapFugCoff+ (AFM./omega_ref)*(vapFugCoff_ref-vapFugCoff);
  H = h + (AFM./omega_ref)*(h - h_ref);
  S = s + (AFM./omega_ref)*(s - s_ref);



//////////////////////////////////// Liquid phase
    //////////////////////////////////////////////
/*
    /========================================================================================================

                                       SIMPLE FLUID Liquid
================================================================================================================
    */
//mixing rules

  AFMl = sum(compMolFrac[2,:].* comp[:].AF);

  VcMl = sum({sum({compMolFrac[2,i].*compMolFrac[2,j].*(1/8)*(1/1000)*((
      comp[i].Vc)^(1/3)+(comp[j].Vc)^(1/3))^3 for j in 1:NOC})for i in 1:
      NOC});

  TcMl = (1/(VcMl^ita))*sum({sum({compMolFrac[2,i].*compMolFrac[2,j
      ].*(((1/8)*(((comp[i].Vc)^(1/3)+(comp[j].Vc)^(1/3))^3)*(1/1000))^(
      ita))*(comp[i].Tc*comp[j].Tc)^(1/2)*BIP_LKP({comp[i].name,comp[j].
      name}) for j in 1:NOC}) for i in 1:NOC});

  PcMl = (0.2905-0.085*AFMl)*R*TcMl./VcMl;

//reduceded mixer property
  Pcrl = P/PcMl;
  Vcrl = Vl/VcMl;
  Tcrl = T/TcMl;

//Eos Equation constant
  Bl = b1[1] - b2[1]./Tcrl - b3[1]./(Tcrl^2) - b4[1]./(Tcrl^3);
  Cl = c1[1] - c2[1]./Tcrl + c3[1]./(Tcrl^3);
  Dl = d1[1] + d2[1]./Tcrl;

```

```
//for volume
  Vcrl = LKP_V(Pcrl, Tcrl, Bl, Cl, Dl, c4[1], bta[1], gma[1], "Liquid");
//for compressibility factor
  Zl = (Pcrl.*Vcrl)./(Tcrl);
//fugacity coefficient
  El = c4[1]/(2*Tcrl^3*gma[1])*(bta[1] +1 - (bta[1] + 1 + gma[1]/Vcrl^2)*
      exp(-gma[1]/Vcrl^2));

  liqFugCoff = Zl - 1 - log(Zl) + Bl/Vcrl + Cl/(2*Vcrl^2) + Dl/(5*Vcrl^5)
       + El;

//enthalpy function
  hl = Tcrl*(Zl - 1 - (b2[1] + 2*b3[1]./Tcrl + 3*b4[1]./Tcrl^2)./(Tcrl*
      Vcrl) - (c2[1]-3*c3[1]./Tcrl^2)/(2*Tcrl*Vcrl^2) + d2[1]./(5*Tcrl*
      Vcrl^5) + 3*El);
//entropy function
  sl = log(Zl) - log(1) - (b1[1] + b3[1]./Tcrl^2 + 2*b4[1]./Tcrl^3)./Vcrl
       -(c1[1]-2*c3[1]./Tcrl^3)./(2* Vcrl^2) - d1[1]/(5*Vcrl^5) + 2*El;

/*
    /==========================================================================

                              REFERANCE FLUID Liquid
==========================================================================/
    */
  Bl_ref = b1[2] - b2[2]./Tcrl - b3[2]./(Tcrl^2) - b4[2]./(Tcrl^3);
  Cl_ref = c1[2] - c2[2]./Tcrl + c3[2]./(Tcrl^3);
  Dl_ref = d1[2] + d2[2]./Tcrl;

//for volume
  Vrl_ref = LKP_V(Pcrl, Tcrl, Bl_ref, Cl_ref, Dl_ref, c4[2], bta[2], gma
      [2], "Liquid");
//for compresibility factor
  Zl_ref = (Pcrl.*Vrl_ref)./(Tcrl);
//fugacity coefficient
  El_ref = c4[2]/(2*Tcrl^3*gma[2])*(bta[2] +1 - (bta[2] + 1 + gma[2]/
      Vrl_ref^2)*exp(-gma[2]/Vrl_ref^2));

  liqFugCoff_ref = Zl_ref - 1 - log(Zl_ref) + Bl_ref/Vrl_ref + Cl_ref/(2*
      Vrl_ref^2) + Dl_ref/(5*Vrl_ref^5) + El_ref;

// enthalpy
  hl_ref = Tcrl*(Zl_ref - 1 - (b2[2] + 2*b3[2]./Tcrl + 3*b4[2]./Tcrl^2)
      ./(Tcrl*Vrl_ref) - (c2[2]-3*c3[2]./Tcrl^2)/(2*Tcrl*Vrl_ref^2) + d2
      [2]./(5*Tcrl*Vrl_ref^5) + 3*El_ref);
//entropy function
  sl_ref = log(Zl_ref) - log(1) - (b1[2] + b3[2]./Tcrl^2 + 2*b4[2]./Tcrl
      ^3)./Vrl_ref -(c1[2]-2*c3[2]./Tcrl^3)./(2* Vrl_ref^2) - d1[2]/(5*
      Vrl_ref^5) + 2*El_ref;

//final equation of gas
  Zfl = Zl + (AFMl./omega_ref)*(Zl_ref -Zl);
  liqFugCofff = liqFugCoff + (AFMl./omega_ref)*(liqFugCoff_ref-liqFugCoff
      );
  Hl = hl + (AFMl./omega_ref)*(hl_ref - hl);
  Sl = sl + (AFMl./omega_ref)*(sl_ref - sl);

```

```

/*
    /==============================================================================
                                        fugacity coefficent for compound
 ==============================================================================/
      */
algorithm
//gas phase
for i in 1:NOC loop
  for j in 1:NOC loop
    sum1[i, j] := 0;
    sum2[i, j] := 0;
    for l in 1:NOC loop
      sum1[i, j] := sum1[i,j] +  compMolFrac[3,l] * (Vcij[l, j] ^ ita *
          Tcij[l, j] - Vcij[l, i] ^ ita * Tcij[l, i]);
      sum2[i, j] :=  sum2[i,j] + compMolFrac[3,l] * (Vcij[l, j] - Vcij[l,
           i]);
    end for;
  end for;
end for;

for i in 1:NOC loop
  for j in 1:NOC loop
    dZ[i, j] := -0.085 * (comp[j].AF - comp[i].AF);
    dV[i, j] := 2 * sum2[i, j];
    dT[i, j] := (2 * sum1[i, j] - 0.25 * VcM ^ (ita - 1) * dV[i, j] * TcM
        ) / VcM ^ ita;
    dP[i, j] := PcM * (dZ[i, j] / (PcM*VcM./(R*TcM)) + dT[i, j] / TcM -
        dV[i, j] / VcM);
  end for;
end for;

for i in 1:NOC loop
  suma[i] := 0;
  sumb[i] := 0;
  sumc[i] := 0;
  for j in 1:NOC loop
    if j<> i then
      suma[i] := suma[i] + compMolFrac[3,j] * dT[i, j];
      sumb[i] := sumb[i] + compMolFrac[3,j] * dP[i, j];
      sumc[i] := sumc[i] + compMolFrac[3,j] * (comp[j].AF - comp[i].AF);
    end if;
  end for;
end for;

for i in 1:NOC loop
  compVapFugCoff[i] := vapFugCofff - 1 / T * H * suma[i] + (Zf - 1) / PcM
       * sumb[i] -(1./omega_ref) *(vapFugCoff_ref-vapFugCoff)* sumc[i];
end for;


algorithm
//liquid phase
for i in 1:NOC loop
  for j in 1:NOC loop
    sum1l[i, j] := 0;
```

```
      sum2l[i, j] := 0;
      for l in 1:NOC loop
        sum1l[i, j] := sum1l[i,j] +  compMolFrac[2,l] * (Vcij[l, j] ^ ita *
              Tcij[l, j] - Vcij[l, i] ^ ita * Tcij[l, i]);
        sum2l[i, j] :=  sum2l[i,j] + compMolFrac[2,l] * (Vcij[l, j] - Vcij[
            l, i]);
      end for;
    end for;
  end for;

  for i in 1:NOC loop
    for j in 1:NOC loop
      dZl[i, j] := -0.085 * (comp[j].AF - comp[i].AF);
      dVl[i, j] := 2 * sum2l[i, j];
      dTl[i, j] := (2 * sum1l[i, j] - ita * VcMl ^ (ita - 1) * dVl[i, j] *
          TcMl) / VcMl ^ ita;
      dPl[i, j] := PcMl * (dZl[i, j] / (PcMl*VcMl./(R*TcMl)) + dTl[i, j] /
          TcMl - dVl[i, j] / VcMl);
    end for;
  end for;

  for i in 1:NOC loop
    sumal[i] := 0;
    sumbl[i] := 0;
    sumcl[i] := 0;
    for j in 1:NOC loop
      if j<> i then
        sumal[i] := sumal[i] + compMolFrac[2,j] * dTl[i, j];
        sumbl[i] := sumbl[i] + compMolFrac[2,j] * dPl[i, j];
        sumcl[i] := sumcl[i] + compMolFrac[2,j] * (comp[j].AF - comp[i].AF)
            ;
      end if;
    end for;
  end for;

  for i in 1:NOC loop
    compLiqFugCoff[i] := liqFugCofff - (1 / T) * Hl * sumal[i] + ((Zfl - 1)
         / PcMl) * sumbl[i] - (1./omega_ref) *(liqFugCoff_ref-liqFugCoff)*
        sumcl[i];
  end for;

  equation
  /*
      /==========================================================================================

                                        K value of compound
  ========================================================================================/
      */
  for i in 1:NOC loop
    if exp(compLiqFugCoff[i]) == 0 or exp(compVapFugCoff[i]) == 0 then
      K[i] = 0;
    else
      K[i] = exp(compLiqFugCoff[i]) / exp(compVapFugCoff[i]);
    end if;
  end for;
  end LEE_KESLER_PLOCKER;
```

Part II

Unifac Automation

Abstract

In previous version of the UNIFAC there is manual input for the unifac subgroup's Q and R value and interaction parameter. In this work I try to develop the OpenModelica model for automation of that manual input. Here you don't find any discussion about the unifac model but only and only discussion about the development of auto input and how to overcome some restriction of Open Modelica and what is the disadvantage of that selected method.

Chapter 3

Data for the UNIFAC model

Hence our data base is based on the ChemSep so for that i also use the ChemSep and the same python script (with some modification) which are used to generate the previous data base.
In ChemSep database unifac data for particular compound are store in the TagName unifacVLE which contain the SubGroupId (We discussion about is letter on) and Value.
See the example given here 3.1.

3.1 ChemSep database

```
<compound>
<LibraryIndex name="Index"  value="509" />
<CompoundID name="Name"  value="N-propylbenzene" />
<StructureFormula name="Structure"  value="(C6H5)CH2CH2CH3" />
<Family name="Family"  value="16" />
<CriticalTemperature name="Critical temperature"
    units="K"  value="638.35" />
<CriticalPressure name="Critical pressure" units="Pa"
    value="3200000" />

... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...

<ChaoSeaderLiquidVolume name="Chao-Seader liquid volume"
    units="m3/kmol"  value="0.139831" />
<UnifacVLE name="UNIFAC" >
  <group id="1"  value="1" />
  <group id="2"  value="2" />
  <group id="10"  value="5" />
  <group id="11"  value="1" />
  </UnifacVLE>
<UnifacLLE name="UNIFAC-LLE" >
  <group id="1"  value="1" />

  ... ... ... ... ... ... ...

</compound>
```

Here use find out that multiple Subgroup are attached with one compound and it also has different different values.
OpenModelica dose not provide as flexibility to pass the different dimension of same type of element so we have to define the length for the array of our unifac data base; and it come out to '5' by python script(Modified python part are given here 3.2)).

3.2 PYTHON script

Here only the part changed by me are shown

```
*** In the XML read part of the python file ***

unifac = [[0 for _ in range(2)]for _ in range(5)]
# size set to 5 after word finding maximum SUBGROUP value
save=[]

*** block given below run for all compound ***
    try:
        unif=comp.getElementsByTagName("UnifacVLE")[0]
        for j in range(5):
            try:
                unifac[j][0] = int(unif.getElementsByTagName("group")
                                        [j].getAttribute("id"))
                unifac[j][1] = int(unif.getElementsByTagName("group")
                                     [j].getAttribute("value"))
                save[i]=j
            except IndexError:
                break
    except IndexError:
        pass

    print(unifac) #print the unifac array value


print(save) # array which contain subgroup value
print(max(save)) #maximum value of subgroup
```

Here So many compounds which do not contain any SubGroup or contain less then 5 subgroup all other vale are zero! which consume unnecessary space.

3.3 Subgroup and R, Q and Aij-Aji value

Subgroup are define as the small part of selected compound and by that subgroup and the value of its repetition can define any compound. 119 such sub group are there which has different R and Q values. This value are tabulated in the 'unifac.txt' file of DWSIM which base file is ChemSep.
All subgroup are also associate with Main Group and such 56 Main group are there in 'unifac.txt' which define the interaction parameter. This interaction parameter are in 'unifac_ip.txt' file. Every main group has 2 interaction parameter with other group known as Aij and Aji.

To extract this data I use python script.

3.4 PYTHON Script

This data file are converted into the '.csv' file for better readability for computer and human.

Python script for RQ value

```
import pandas as pd

RQ = pd.read_csv('unifacrq.csv')

sub_group_id = RQ['SUB_ID']
main_group_id = RQ['ID']

main_group_name = RQ['Maingroups']
sub_group_name = RQ['Group']

Rk = RQ['Rk']
Qk = RQ['Qk']

matrix2 = [[0 for _ in range(2)]for _ in range(len(Rk))]

j = 1
k = 1
for i in range(len(sub_group_id)):
    matrix2[i][0]=Rk[i]
    matrix2[i][1]=Qk[i]

matrix_name2 = [[0 for _ in range(3)] for _ in range(len(sub_group_id))]
for i in range(len(sub_group_id)):
    matrix_name2[i][0]= i+1
    matrix_name2[i][1]= str(main_group_name[i])
    matrix_name2[i][2]= str(sub_group_name[i])


f = open("UNIFAC_RQ_119.txt","+w")
f.write(str(matrix2))
f.close()
f = open("UNIFAC_RQ_name_119.txt","+w")
f.write(str(matrix_name2))
f.close()

f = open("ID.txt","+w")
for i in range(0,119):
    f.write(str(str(main_group_id[i])+',  '))
f.close()
print(main_group_id)
print(len(main_group_id))
input()
```

Python script for binary interaction parameter

```
import pandas as pd


IP =  pd.read_csv('unifac_ip.csv')

group_i = IP['group_i']
group_j = IP['group_j']

group_name = IP['id_i']

AIJ = IP['aij']
AJI = IP['aji']


Matric = [[0 for x in range(56)] for y in range(56)]

for i in range(0,len(AIJ)):
    Matric[group_i[i]-1][group_j[i]-1]=AIJ[i]
    Matric[group_j[i]-1][group_i[i]-1] = AJI[i]

name = [[0 for x in range(2)] for y in range(56)]
id = group_name[0]
j=0
l=0
for k in range(len(group_name)):
    id = group_name[j]
    name[l][0] = l+1
    name[l][1] = group_name[j]
    if id != group_name[k]:
        j=k
        l=l+1


Matric2 = [[0 for x in range(2)] for y in range(len(AIJ))]
Matric3 = [[0 for x in range(2)] for y in range(len(AIJ))]


for i in range(len(AIJ)):
    Matric2[i][0]=group_i[i]
    Matric2[i][1]=group_j[i]

    Matric3[i][0]=AIJ[i]
    Matric3[i][1]=AJI[i]

f= open("unifacBIP.txt","w+")
f.write(str(Matric))
f.close()

f2 = open("unifacID.txt","w+")
f2.write(str(name))
f2.close()

f= open("unifacBIP2.txt","w+")
f.write(str(Matric2))
f.close()

f2 = open("unifacID2.txt","w+")
```

```
f2.write(str(Matric3))
f2.close()

print(Matric)
print(name)
print(Matric3)
print(Matric2)
input()
```

Here if we use the 56X56 matrix with mapping then the compilation time or reading time going to be very less instead of searching from 1400X2 matrix; but the size of data is reduced in 1400X2. here is use the 56X56 matrix with direct mapping.

Chapter 4

Flow of Model

In modelica there is one restriction in the array declaration; We have to define the size of array before the compiling. But here the size of unique subgroup is going to change from compounds to compounds.
To solve this problem I use the functions loop in OpenModelica which first calculate length with from given compounds and ‘UNIFAC_SubGroup’ then by this length and compounds ‘UNIFAC_SubGroup’ it calculate unique Sub Id array and then by length, unique Sub ID array it calculate the gamma with help of Binary Interaction parameter and R-Q values function.
All function return only gamma, gammadew, gammabubl value which are array with length of number of compound; Except Binary Interaction parameter function and R-Q value function, they return the Binary Interaction parameter function and R-Q value simultaneously with help of ‘UNIFAC_SubGroup’ and length of unique Sub ID.
Overview of the data flow in the model are shown in the figure 4.1. In this flow all calculation related to unifac are calculated in 4th function, UNIFAC_gamma.
Algorithm for finding the length and unique id array are same, flowchart for the same are shown in the figure 4.2.

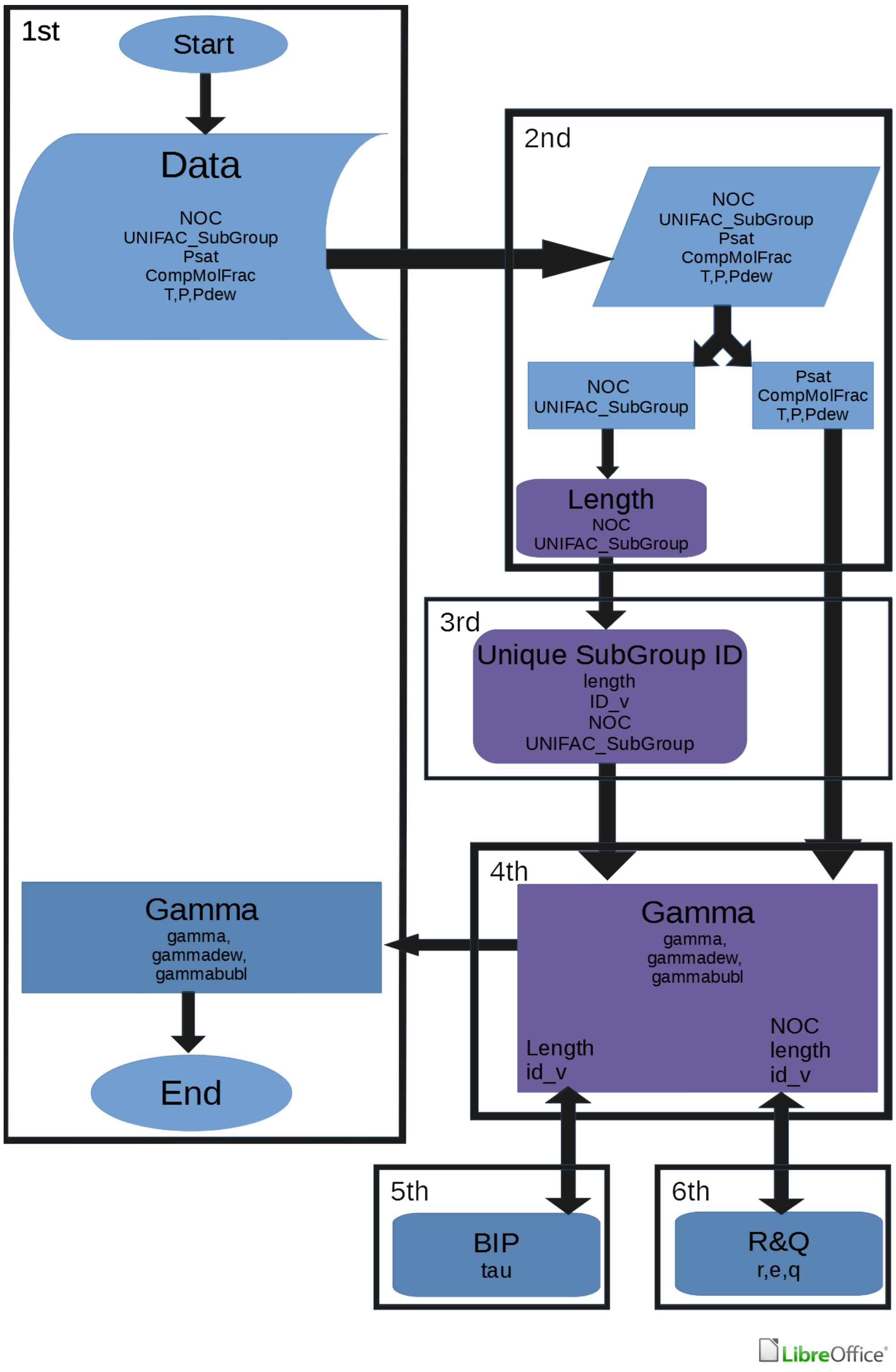

Figure 4.1: Data Flow in UNIFAC model

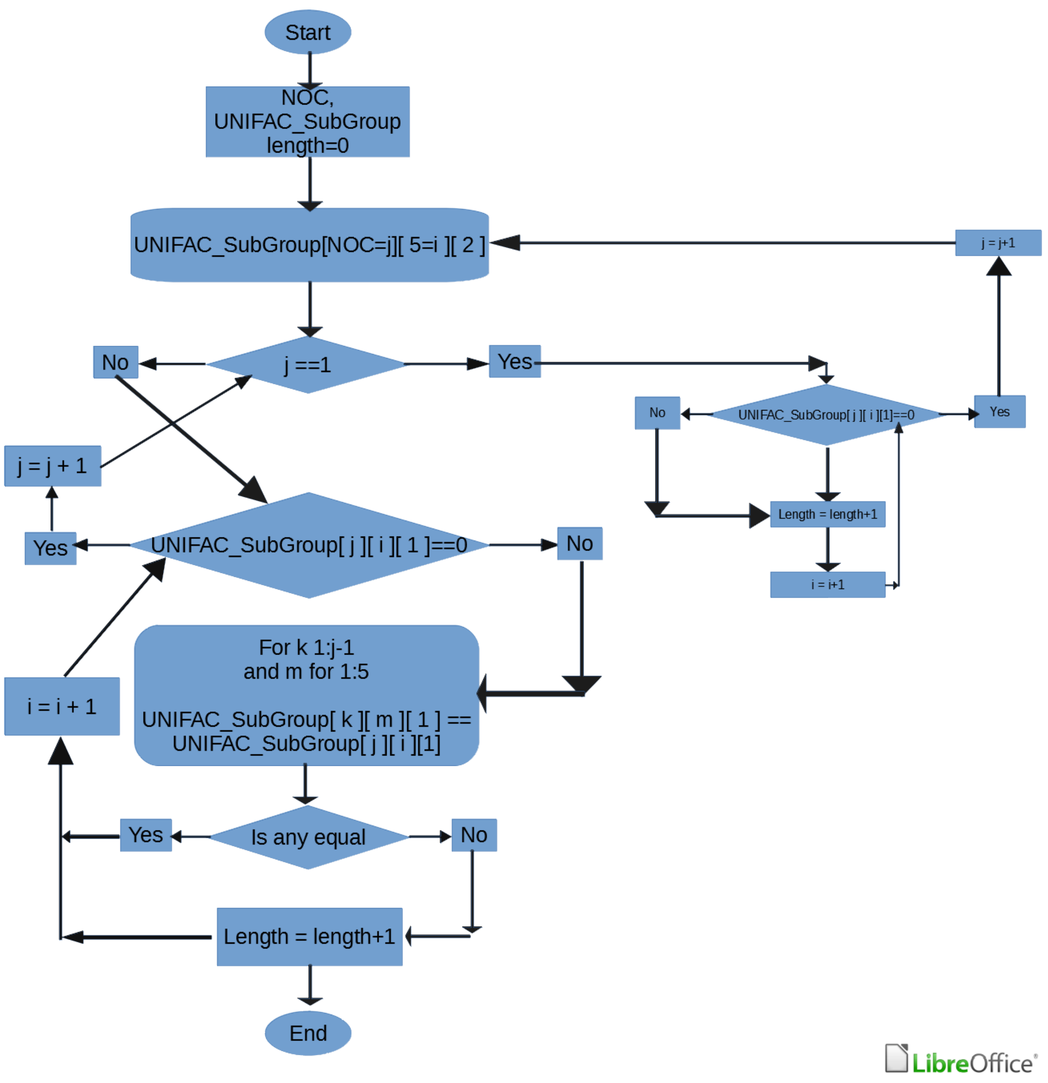

Figure 4.2: Flowchart of finding length

4.1 Code

UNIFAC model (1st in figure 4.1)

```
model UNIFAC
     import data = Chemsep_Database;
     parameter data.Methylethylketone meth;
     parameter data.Aceticacid eth;
//Instantiation of selected compounds
     parameter Chemsep_Database.General_Properties comp[NOC] = {meth, eth
         };
     parameter Integer NOC = 2 "Number of components";
     Real T=298.15;
```

```
    Real P=101325;
    Real compMolFrac[3,NOC]={{0.5,0.5},{0.5,0.5},{0.5,0.5}};
    Real Psat[NOC];
    Real gamma[NOC];
    Real K[NOC];
//Activity Coefficient at the Bubble and Dew Points
    Real gammaBubl[NOC], gammaDew[NOC](each start = 1.5);

equation
    for i in 1:NOC loop
      Psat[i] = Simulator.Files.Thermodynamic_Functions.Psat(comp[i].VP
          [:], 298);
    end for;
      (gamma,gammaBubl,gammaDew)=UNIFAC_M(
          NOC,comp.UNIFAC_SubGroup,Psat,T,P,compMolFrac);

    for i in 1:NOC loop
      K[i] = gamma[i] * Psat[i] / P;
    end for;
end UNIFAC;
```

UNIFAC_M functions in Loop (2nd in figure 4.1)

```
function UNIFAC_M
import dat = unifac;
  input Integer NOC;
  input Integer ID[NOC, 5, 2];
  input Real Psat[NOC];
  input Real T;
  input Real P;
  input Real X[3,NOC];
  output Real gamma[NOC];
  output Real gammaBubl[NOC];
  output Real gammaDew[NOC];
protected
  parameter Integer N = 5;
  Integer length;
  Real i, j, k, l, str;
algorithm
  length := 0;
  for i in 1:NOC loop
    str := 0;
    if i == 1 then
      for j in 1:N loop
        if ID[i, j, 1] <> 0 then
          length := length + 1;
        end if;
      end for;
    else
      for j in 1:N loop
        if ID[i,j,1]==0 then
          break;
        end if;
        for k in 1:i - 1 loop
          for l in 1:N loop
            if ID[k,l,1]==0 then
```

```
              break;
            elseif ID[i, j, 1] == ID[k, l, 1] then
              str := 1;
            end if;
          end for;
        end for;
        if str == 0 then
          length := length + 1;
        end if;
        str := 0;
      end for;
    end if;
  end for;

  (gamma,gammaBubl,gammaDew):=UNIFAC_ID(NOC,ID,length,Psat,T,P,X);
end UNIFAC_M;
```

UNIFAC_ID function (3rd in figure 4.1)

```
function UNIFAC_ID
import dat = unifac;
  input Integer NOC;
  input Integer ID[NOC, 5, 2];
  input Integer length;
  input Real Psat[NOC];
  input Real T;
  input Real P;
  input Real X[3,NOC];
  output Real gamma[NOC];
  output Real gammaBubl[NOC];
  output Real gammaDew[NOC];
protected
  Integer ID_v[length];
  Integer i, j, k, l, str;
  Integer v;
  parameter Integer N = 4;

algorithm
v:=1;
for i in 1:NOC loop
    str := 0;
    if i == 1 then
      for j in 1:N loop
        if ID[i, j, 1] <> 0 then
          ID_v[v]:=ID[i, j, 1];
          v := v + 1;
        end if;
      end for;
    else
      for j in 1:N loop
        if ID[i,j,1]==0 then
          break;
        end if;
        for k in 1:i - 1 loop
          for l in 1:N loop
            if ID[k,l,1]==0 then
```

```
              break;
            elseif ID[i, j, 1] == ID[k, l, 1] then
              str := 1;
            end if;
          end for;
        end for;
        if str == 0 then
          ID_v[v]:=ID[i, j, 1];
          v := v + 1;
        end if;
        str := 0;
      end for;
    end if;
  end for;

  (gamma,gammaBubl,gammaDew):=UNIFAC_gamma(NOC,length,ID,ID_v,Psat,T,P,X)
      ;
end UNIFAC_ID;
```

UNIFAC_gamma function (4th in figure 4.1)

```
function UNIFAC_gamma
  input Integer NOC;
  input Integer length;
  input Integer ID[NOC, 5, 2];
  input Integer ID_v[length];
  input Real Psat[NOC];
  input Real T;
  input Real P;
  input Real X[3,NOC];
  output Real gammac[NOC];
  output Real gammaBubl[NOC];
  output Real gammaDew[NOC];
  protected
  //Intermediate values used to compute UNIFAC R and Q values
  Real q[NOC] "Van der walls molecular surface area";
  Real r[NOC] "Van der walls molecular volume";
  Real e[length, NOC] "Group Surface area fraction of comp i";
  Real tau[length, length] "Boltzmann factors";
  Real B[NOC, length] "UNIFAC parameter ";
  Real theta[length] "UNIFAC parameter";
  Real sum[NOC];
  Real S[length] "Unifac parameter ";
  Real J[NOC]
"Surface area fraction of comp i";
  Real L[NOC] "Molecular volume fraction of comp i";
  //Activity Coefficients
  Real gammar[NOC] "Residual activity coefficient of comp i";
  Real liqfugcoeff_bubl[NOC], vapfugcoeff_dew[NOC];
  //Excess Energy Properties
  Real resMolSpHeat[3], resMolEnth[3], resMolEntr[3];
  //
      ================================================================================

  //Bubble Point Calculation Variables
  Real theta_bubl[length] "UNIFAC parameter";
  Real S_bubl[length] "Unifac parameter ";
  Real J_bubl[NOC] "Surface area fraction of comp i";
  Real L_bubl[NOC] "Molecular volume fraction of comp i";
  Real gammac_bubl[NOC] "Combinatorial activity coefficient of components at bubble point";
  Real gammar_bubl[NOC] "Residual activity coefficient of components at bubble point";
  Real sum_bubl[NOC];
  //==================================================================================
  //Dew Point Calculation Routine
  Real theta_dew[length] "UNIFAC parameter";
  Real S_dew[length] "Unifac parameter ";
  Real J_dew[NOC] "Surface area fraction of comp i";
  Real L_dew[NOC] "Molecular volume fraction of comp i";
  Real gammac_dew[NOC] "combinatorial activity coefficient of components at dew point";
  Real gammar_dew[NOC] "residual activity coefficient of components at dew point";
  Real sum_dew[NOC];
  Real dewLiqMolFrac[NOC](each start = 0.5);
algorithm

  tau := UNIFAC_BIP(length,ID_v,T);
  (r,q,e) := UNIFAC_RQ(NOC,ID);

  for i in 1:NOC loop
    J[i] := r[i] / sum(r[:] .* X[2, :]);
    L[i] := q[i] / sum(q[:] .* X[2, :]);
    gammac[i] := exp(1 - J[i] + log(J[i]) + (-5 * q[i] * (1 - J[i] / L[i] + log(J[i] / L[i]))));
  end for;

  for j in 1:length loop
    theta[j] := sum(compMolFrac[2, :] .* q[:] .* e[j, :]) / sum(X[2, :] .* q[:]);
  end for;
  for i in 1:length loop
    S[i] := sum(theta[:] .* tau[:, i]);
  end for;

  for i in 1:NOC loop
    for j in 1:length loop
      for l in 1:m loop
        B[i, j] := sum(e[:, i] .* tau[:, j]);
      end for;
    end for;
  end for;
  sum[:] := fill(0, NOC);
  for j in 1:length loop
    for i in 1:NOC loop
      sum[i] := sum[i] + theta[j] * B[i, j] / S[j] - e[j, i] * log(B[i, j] / S[j]);
      gammar[i] := exp(q[i] * (1 - sum[i]));
    end for;
  end for;

  for i in 1:NOC loop
    gamma[i] := exp(log(gammar[i]) + log(gammac[i]));
  end for;

  for i in 1:NOC loop
    J_bubl[i] := r[i] / sum(r[:] .* X[1, :]);
    L_bubl[i] := q[i] / sum(q[:] .* X[1, :]);
    gammac_bubl[i] := exp(1 - J_bubl[i] + log(J_bubl[i]) + (-5 * q[i] *
        (1 - J_bubl[i] / L_bubl[i] + log(J_bubl[i] / L_bubl[i]))));
  end for;

  for j in 1:length loop
    theta_bubl[j] := sum(X[1, :] .* q[:] .* e[j, :]) / sum(X[1, :] .* q
        [:]);
  end for;
  for i in 1:length loop
    S_bubl[i] := sum(theta_bubl[:] .* tau[:, i]);
  end for;

  sum_bubl[:] := fill(0, NOC);
  for j in 1:length loop
    for i in 1:NOC loop
      sum_bubl[i] := sum_bubl[i] + theta_bubl[j] * B[i, j] / S_bubl[j] -
          e[j, i] * log(B[i, j] / S_bubl[j]);
      gammar_bubl[i] := exp(q[i] * (1 - sum_bubl[i]));
    end for;
  end for;


  for i in 1:NOC loop
    gammaBubl[i] := exp(log(gammar_bubl[i]) + log(gammac_bubl[i]));
  end for;
//
    ============================================================================================================
//Dew Point Calculation Routine
  for i in 1:NOC loop
    dewLiqMolFrac[i] := compMolFrac[1, i] * Pdew / (gammaDew[i] * Psat[i
        ]);
  end for;

  for i in 1:NOC loop
    J_dew[i] := r[i] / sum(r[:] .* dewLiqMolFrac[:]);
    L_dew[i] := q[i] / sum(q[:] .* dewLiqMolFrac[:]);
    gammac_dew[i] := exp(1 - J_dew[i] + log(J_dew[i]) + (-5 * q[i] * (1 -
         J_dew[i] / L_dew[i] + log(J_dew[i] / L_dew[i]))));
  end for;

  for j in 1:length loop
    theta_dew[j] := sum(dewLiqMolFrac[:] .* q[:] .* e[j, :]) / sum(
        dewLiqMolFrac[:] .* q[:]);
  end for;
  for i in 1:length loop
    S_dew[i] := sum(theta_dew[:] .* tau[:, i]);
  end for;

```

```
    sum_dew[:] := fill(0, NOC);
    for j in 1:length loop
      for i in 1:NOC loop
        sum_dew[i] := sum_dew[i] + theta_dew[j] * B[i, j] / S_dew[j] - e[j,
             i] * log(B[i, j] / S_dew[j]);
        gammar_dew[i] := exp(q[i] * (1 - sum_dew[i]));
      end for;
    end for;

    for i in 1:NOC loop
      gammaDew[i] := exp(log(gammar_dew[i]) + log(gammac_dew[i]));
    end for;

  end UNIFAC_gamma;
```

UNIFAC_BIP function (5th in figure 4.1)

```
  function UNIFAC_BIP
    input Integer length;
    input Integer ID[length];
    input Real T;
    output Real tau[length, length] "Boltzmann factors";
    protected
    Real A[length,length];
    parameter Integer Main_ID[119] = {1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 4, 4,
        4, 5, 6, 7, 8, 9, 9, 10, 11, 11, 12, 13, 13, 13, 13, 14, 14, 14,
        15, 15, 15, 16, 16, 17, 18, 18, 18, 19, 19, 20, 20, 21, 21, 21, 22,
         22, 22, 23, 23, 24, 25, 26, 26, 26, 27, 28, 29, 29, 30, 31, 32,
        33, 34, 34, 35, 36, 37, 38, 39, 39, 40, 40, 40, 41, 42, 42, 42, 42,
         43, 43, 43, 44, 45, 45, 45, 45, 45, 45, 45, 45, 46, 46, 46, 46,
        46, 46, 47, 47, 48, 48, 48, 49, 50, 50, 50, 52, 52, 53, 53, 53, 53,
         53, 53, 54, 55, 56};
    //ID number and name
    //[[1, 'CH2'], [2, 'C=C'], [3, 'ACH'], [4, 'ACCH2'], [5, 'OH'], [6, '
        MeOH'], [7, 'H2O'], [8, 'ACOH'], [9, 'CH2CO'], [10, 'CHO'], [11, '
        COOC'], [12, 'HCOO'], [13, 'CH2O'], [14, 'CNH2'], [15, 'CNH'], [16,
         'C3N'], [17, 'ACNH2'], [18, 'Pyridine'], [19, 'CCN'], [20, 'COOH
        '], [21, 'CCl'], [22, 'CCl2'], [23, 'CCl3'], [24, 'CCl4'], [25, '
        ACCl'], [26, 'CNO2'], [27, 'ACNO2'], [28, 'CS2'], [29, 'CH3SH'],
        [30, 'Furfural'], [31, 'DOH'], [32, 'I'], [33, 'Br'], [34, 'C#C'],
        [35, 'DMSO'], [36, 'ACRY'], [37, 'ClCC'], [38, 'ACF'], [39, 'DMF'],
         [40, 'CF2'], [41, 'COO'], [42, 'SiH2'], [43, 'SiO'], [44, 'NMP'],
        [45, 'CClF'], [46, 'CON'], [47, 'OCCOH'], [48, 'CH2S'], [49, '
        Morpholine'], [50, 'Thiophene'],[51, NaN] [52, 'CH2SuCH2'], [53, '
        Oxides'], [54, 'Anhydride'], [55, 'Aromatic Nitrile'], [56, '
        Aromatic Bromo']]
    parameter Real BIP[56,56]={{0.0, 86.02, 61.13, 76.5, 986.5, 697.2,
        1318.0, 1333.0, 476.4, 677.0, 232.1, 507.0, 251.5, 391.5, 255.7,
        206.6, 920.7, 287.77, 597.0, 663.5, 35.93, 53.76, 24.9, 104.3,
        11.44, 661.5, 543.0, 153.6, 184.4, 354.55, 3025.0, 335.8, 479.5,
        298.9, 526.5, 689.0, -4.189, 177.12, 485.3, -2.859, 387.1, -244.59,
         745.3, 0.0, 0.0, 0.0, 0.0, 187.0, 216.1, 92.99, 0, 808.59, 408.3,
        718.01, 0, 153.72}, {-35.36, 0.0, 38.81, 74.15, 524.1, 787.6,
        270.6, 526.1, 182.6, 448.8, 37.85, 333.5, 214.5, 240.9, 163.9,
  61.11, 749.3, 280.5, 336.9, 318.9, -36.87, 58.55, -13.99, -109.7, 100.1,
      357.5, 0, 76.30199999999999, 0, 262.9, 0, 0, 183.8, 31.14, 179.0,
      -52.87, -66.46, 125.8, -70.45, 449.4, 48.33, 0, 0, 220.3, 0, 390.9,
      553.3, -617.0, 62.56, 0, 0, 200.94, 219.9, -677.25, 0, 0}, {-11.12,
      3.446, 0.0, 167.0, 636.1, 637.35, 903.8, 1329.0, 25.77, 347.3, 5.994,
       287.1, 32.14, 161.7, 122.8, 90.49, 648.2, -4.449, 212.5, 537.4,
      -18.81, -144.4, -231.9, 3.0, 187.0, 168.0, 194.9, 52.07, -10.43,
      -64.69, 210.4, 113.3, 261.3, 154.26,
  169.9, 383.9, -259.1, 359.3, 245.6, 22.67, 103.5, -450.4, 252.7, 86.46,
      -5.869, 0, 268.1, 0, -59.58, -39.16, 0, 360.82, 171.49, 272.33,
      22.06, 174.35}, {-69.7, -113.6, -146.8, 0.0, 803.2, 603.25, 5695.0,
      884.9, -52.1, 586.6, 5688.0, 197.8, 213.1, 19.02, -49.29, 23.5,
      664.2, 52.8, 6096.0, 872.3, -114.1, -111.0, -80.25, -141.3, -211.0,
      3629.0, 4448.0, -9.451, 393.6, 48.49, 4975.0, 259.0, 210.0, -152.55,
      4284.0, -119.2, -282.5, 389.3, 5629.0, -245.39, 69.26, -432.3, 238.9,
       30.04, 0, 0, 333.3, 0, -203.6,
  184.9, 0, 233.51, -184.68, 9.63, 795.38, -280.9}, {156.4, 457.0, 89.6,
      25.82, 0.0, -137.1, 353.5, -259.7, 84.0, -203.6, 101.1, 267.8, 28.06,
       83.02, 42.7, -323.0, -52.39, 170.0, 6.712000000000001, 199.0, 75.62,
       65.28, -98.12, 143.1, 123.5, 256.5, 157.1, 488.9, 147.5, -120.5,
      -318.9, 313.5, 202.1, 727.8, -202.1, 74.27, 225.8, 101.4, -143.9, 0,
      190.3, 683.3, 355.5, 46.38, -88.11, 200.2, 421.9, 0, 104.7, 57.65, 0,
       215.81, 6.39, 0, 0, 147.97}, {16.51, -12.52, -50.0, -44.5, 249.1,
      0.0, -181.0, -101.7, 23.39,
  306.4, -10.72, 179.7, -128.6, 359.3, -20.98, 53.9, 489.7, 580.5, 53.28,
      -202.0, -38.32, -102.5, -139.4, -44.76, -28.25, 75.14, 457.88,
      -31.09, 17.5, -61.76, -119.2, 212.1, 106.3, -119.1, -399.3, -5.224,
      33.47, 44.78, -172.4, 0, 165.7, 0, 0, 0, 72.96, 0, 0, 37.63, -59.4,
      -46.01, 0, 150.02, 98.2, 0, 0, 0}, {300.0, 496.1, 362.3, 377.6,
      -229.1, 289.6, 0.0, 324.5, -195.4, -116.0, 72.87, 233.87, 540.5,
      48.89, 168.0, 304.0, 459.0, 459.0, 112.6, -14.09, 325.4, 370.4,
      353.7, 497.5, 133.9, 220.6, 399.5, 887.1,
  0, 188.0, 12.72, 0, 777.1, 0, -139.0, 160.8, 0, 0, 319.0, 0, -197.5,
      -817.7, 0, -504.2, 0, -382.7, -248.3, 0, 407.9, 0, 0, -255.63,
      -144.77, 0, 0, 580.28}, {275.8, 217.5, 25.34, 244.2, -451.6, -265.2,
      -601.8, 0.0, -356.1, -271.1, -449.4, -32.52, -162.9, -832.97, 0, 0,
      -305.5, -305.5, 0, 408.9, 0, 517.27, 0, 1827.0, 6915.0, 0, -413.48,
      8484.0, 0, 0, -687.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -494.2, 0, 0,
      -452.2, 0, 0, 0, 0, 0, 1005.0, 0, 0, 0, 0, 0, 0}, {26.76, 42.92,
      140.1, 365.8, 164.5, 108.7, 472.5, -133.1, 0.0,
  -37.36, -213.7, -190.4, -103.6, 0, -174.2, -169.0, 6201.0, 7.341, 481.7,
      669.4, -191.7, -130.3, -354.6, -39.2, -119.8, 137.5, 548.5, 216.1,
      -46.28, -163.7, 71.46, 53.59, 245.2, -246.6, -44.58, -63.5, -34.57,
      0, -61.7, 0, -18.8, -363.8, 0, 0, 0, 0, 139.6, 0, 0, -162.6, 0, 0,
      -288.94, 91.01, 0, 179.74}, {505.7, 56.3, 23.39, 106.0, 529.0,
      -340.2, 480.8, -155.6, 128.0, 0.0, -110.3, 766.0, 304.1, 0, 0, 0, 0,
      0, -106.4, 497.5, 751.9, 67.52, -483.7, 0, 0, 0, 0, 0, 0, 0, 0,
      117.0, 0, 2.21, 0, -339.2, 172.4, 0, -268.8, 0, -275.5, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 79.71, 0, 0, 0}, {114.8, 132.1, 85.84, -170.0,
      245.4,
  249.63, 200.8, -36.72, 372.2, 185.1, 0.0, -241.8, -235.7, 0, -73.5,
      -196.7, 475.5, -0.13, 494.6, 660.2, -34.74, 108.9, -209.7, 54.57,
      442.4, -81.13, 0, 183.0, 0, 202.3, -101.7, 148.3, 18.88, 71.48,
      52.08, -28.61, -275.2, 0, 85.33, 0, 560.2, 0, 0, 0, 0, 0, 37.54, 0,
      0, 0, 0, 0, 36.34, 446.9, 0, 0}, {329.3, 110.4, 18.12, 428.0, 139.4,
      227.8, 124.63, -234.25, 385.4, -236.5, 1167.0, 0.0, -234.0, 0, 0, 0,
      0, -233.4, -47.25, -268.1, 0, 31.0, -126.2, 179.7, 24.28, 0, 0, 0,
      103.9, 0, 0, 0, 298.13, 0, 0, 0,
  -11.4, 0, 308.9, 0, -70.24, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -77.96, 0,
  0, 0}, {83.36, 26.51, 52.13, 65.69, 237.7, 238.4, -314.7, -178.5,
      191.1, -7.837999999999999, 461.3, 457.3, 0.0, -78.36, 251.5, 5422.3,
      -46.39, 213.2, -18.51, 664.6, 301.1, 137.8, -154.3, 47.67, 134.8,
      95.18, 155.11, 140.9, -8.538, 170.1, -20.11, -149.5, -202.3, -156.57,
       128.8, 0, 240.2, -48.25, 254.8, -172.51, 417.0, -588.9, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 567.0, 102.21, 0, 0}, {-30.48, 1.163, -44.85,
      296.4, -242.8, -481.7, -330.48,
  -870.8, 0, 0, 0, 0, 222.1, 0.0, -107.2, -41.11, -200.7, 0, 358.9, 0,
      -82.92, 0, 0, -99.81, 30.05, 0, 0, 0, -70.14, 0, 0, 0, 0, 0, 874.19,
      0, 0, 0, -164.0, 0, 0, 1338.0, 202.7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0}, {65.33, -28.7, -22.31, 223.0, -150.0, -370.3, -448.2, 0,
      394.6, 0, 136.0, 0, -56.08, 127.4, 0.0, -189.2, 138.54, 431.49,
      147.1, 0, 0, 0, 0, 71.23, -18.93, 0, 0, 0, 0, 0, 939.07, 0, 0, 0, 0,
      0, 0, -273.9, -255.22, 0, -38.77, -664.4, 275.9, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0}, {-83.98, -25.38,
  -223.9, 109.9, 28.6, -406.8, -598.8, 0, 225.3, 0, 2889.0, 0, -194.1,
      38.89, 865.9, 0.0, 287.43, 0, 1255.1, 0, -182.91, -73.85, -352.9,
      -262.0, -181.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 243.1, 0, 0, 570.9, 22.05,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {1139.0, 2000.0,
       247.5, 762.8, -17.4, -118.1, -341.6, -253.1, -450.3, 0, -294.8, 0,
      285.36, -15.07, 64.3, -24.46, 0.0, 89.7, -281.6, -396.0, 287.0,
      -111.0, 0, 882.0, 617.5, 0, -139.3, 0, 0, 0, 0.1004, 0, 0, 0, 0, 0,
      0, 0, -334.4, 0, -89.42, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {-101.6, -47.63, 31.87, 49.8,
      -132.3, -378.2, -332.9, -341.6, 29.1, 0, 8.87, 554.4, -156.1, 0,
      -207.66, 0, 117.4, 0.0, -169.7, -153.7, 0, -351.6, -114.7, -205.3,
      -2.17, 0, 2845.0, 0, 0, 0, 0, 0, -60.78, 0, 0, 0, 160.7, -196.3, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, -136.6, 0, 0, 0, 98.82, 0, 0}, {24.82,
      -40.62, -22.97, -138.4, 185.4, 162.6, 242.8, 0, -287.5, 224.66,
      -266.6, 99.37, 38.81, -157.3, -108.5, -446.86, 777.4, 134.3, 0.0,
      205.27, 4.933, -152.7, -15.62, -54.86, -4.624, -0.515, 0, 230.9,
      0.4604, 0, 177.5, 0, -62.17, -203.0, 0, 81.57, -55.77, 0, -151.5, 0,
  120.3, 0, 0, 0, 0, 0, 151.8, 0, 0, 0, 0, 0, 0, 0, 0}, {315.3, 1264.0,
      62.32, 89.86, -151.0, 339.8, -66.17, -11.0, -297.8, -165.5, -256.3,
      193.9, -338.5, 0, 0, 0, 493.8, -313.5, 92.07, 0.0, 13.41, -44.7,
      39.63, 183.4, -79.08, 0, 0, 0, 0, -208.9, 0, 228.4, -95.0, 0, -463.6,
       0, -11.16, 0, -228.0, 0, -337.0, 448.1, -1327.0, 0, 0, 835.6, 0, 0,
      0, 0, 0, 0, 12.55, -60.07, 88.09, 0}, {91.46, 40.25, 4.68, 122.9,
      562.2, 529.0, 698.2, 0, 286.3, -47.51, 35.38, 0, 225.4, 131.2, 0,
      151.38, 429.7, 0, 54.32, 519.1, 0.0, 108.3, 249.2, 62.42, 153.0,
      32.73, 86.2, 450.1, 59.02, 65.56, 0, 2.22, 344.4, 0, 0, 0, -168.2, 0,
  6.57, 0, 63.67, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -127.9, 0, 0, 0},
      {34.01, -23.5, 121.3, 140.8, 527.6, 669.9, 708.7, 1633.5, 82.86,
      190.6, -132.9, 80.99, -197.7, 0, 0, -141.4, 140.8, 587.3, 258.6,
      543.3, -84.53, 0.0, 0.0, 56.33, 223.1, 108.9, 0, 0, 0, 149.56, 0,
      177.6, 315.9, 0, 215.0, 0, -91.8, 0, -160.28, 0, -96.87, 0, 0, 0, 0,
      0, 16.23, 0, 0, 0, 0, 0, 0, 0, 0}, {36.7, 51.06, 288.5, 69.9,
      742.1, 649.1, 826.76, 0, 552.1, 242.8, 176.5, 235.6, -20.93, 0, 0,
      -293.7, 0, 18.98, 74.04, 504.2,
  -157.1, 0.0, 0.0, -30.1, 192.1, 0, 0, 116.6, 0, -64.38, 0, 86.4, 168.8,
      0, 363.7, 0, 111.2, 0, 0, 0, 255.8, 0, 0, -659.0, 0, 0, 0, 565.9, 0,
      0, 0, 0, 165.67, 0, 0, 0}, {-78.45, 160.9, -4.7, 134.7, 856.3, 709.6,
       1201.0, 10000.0, 372.0, 0, 129.5, 351.9, 113.9, 261.1, 91.13, 316.9,
       898.2, 368.5, 492.0, 631.0, 11.8, 17.97, 51.9, 0.0, -75.97, 490.9,
      534.7, 132.2, 0, 546.7, 0, 247.8, 146.6, 0, 337.7, 369.5, 187.1,
      -158.8, 498.6, 0, 256.5, 0, 127.2, 0, 0, 0, 361.1, 63.95, 0, 108.5,
      0, 585.19, 291.87, 532.73, 0, 127.16}, {106.8, 70.32, -97.27, 402.5,
      325.7, 612.8, −274.5, 622.3, 518.4, 0, −171.1, 383.3,
 −25.15, 108.5, 102.2, 2951.0, 334.9, 20.18, 363.5, 993.4, −129.7, −8.309,
       −0.2266, −248.4, 0.0, 132.7, 2213.0, 0, 0, 0, 0, 0, 593.4, 0,
      1337.37, 0, 0, 0, 5143.14, 309.58, −145.1, 0, 0, −35.68, 0, 0, 423.1,
       0, 0, 0, 0, 0, 0, 0, 0, 8.48}, {−32.69, −1.996, 10.38, −97.05,
      261.6, 252.6, 417.9, 0, −142.6, 0, 129.3, 0, −94.49, 0, 0, 0, 0, 0,
      0.2827, 0, 113.0, −9.639, 0, −34.68, 132.9, 0.0, 533.2, 320.2, 0, 0,
      139.8, 304.3, 10.17, −27.7, 0, 0, 10.76, 0, −223.1, 0, 248.4, 0, 0,
      0, −52.1, 0, 0, 0, 0,
 −4.565, 0, 0, 0, 0, 0, 0}, {5541.0, 0, 1824.0, −127.8, 561.6, 511.29,
      360.7, 815.12, −101.5, 0, 0, 0, 220.66, 0, 0, 0, 134.9, 2475.0, 0, 0,
       1971.0, 0, 0, 514.6, −123.1, −85.12, 0.0, 0, 0, 0, 0, 2990.0,
      −124.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 1742.53}, {−52.65, 16.62, 21.5, 40.68, 609.8, 914.2, 1081.0,
      1421.0, 303.7, 0, 243.8, 0, 112.4, 0, 0, 0, 0, 0, 335.7, 0, −73.09,
      0, −26.06, −60.71, 0, 277.8, 0, 0.0, 0, 0, 0, 292.7, 0, 0, 0, 0,
      −47.37, 0, 0, 0, 469.8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 684.78,
      0, 0}, {−7.481, 0, 28.41, 19.56, 461.6, 448.6, 0, 0, 160.6, 0, 0,
 201.5, 63.71, 106.7, 0, 0, 0, 0, 161.0, 0, −27.94, 0, 0, 0, 0, 0, 0, 0,
      0.0, 0, 0, 0, 0, 0, 31.66, 0, 0, 0, 78.92, 0, 0, 0, 0, −209.7, 0, 0,
      0, −18.27, 0, 0, 0, 0, 0, 0, 0}, {−25.31, 82.64, 157.3, 128.8,
      521.6, 287.0, 23.48, 0, 317.5, 0, −146.3, 0, −87.31, 0, 0, 0, 0, 0,
      0, 570.6, −39.46, −116.21, 48.48, −133.16, 0, 0, 0, 0, 0, 0.0, 0, 0,
      0, 0, 0, 0, 262.9, 0, 0, 0, 43.37, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0}, {140.0, 0, 221.4, 150.6, 267.6, 240.8, −137.4, 838.4,
      135.4, 0, 152.0, 0, 9.207, 0, −213.74, 0, 192.3, 0, 169.6, 0, 0, 0,
      0, 0, 0, 481.3, 0, 0, 0, 0, 0.0, 0, 0, 0, −417.2, 0, 0, 0, 302.2, 0,
      347.8, 0, 0, 1004.0, 0, 0, 434.1, 0, 0, 0, 0, 0, 0, 0, 0}, {128.0,
       0, 58.68, 26.41,
 501.3, 431.3, 0, 0, 138.0, 245.9, 21.92, 0, 476.6, 0, 0, 0, 0, 0, 0,
      616.6, 179.25, −40.82, 21.76, 48.49, 0, 64.28, 2448.0, −27.45, 0, 0,
      0, 0.0, 6.37, 0, 0, 0, 0, 0, 0, 0, 68.55, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 190.81, 0, 0}, {−31.52, 174.6, −154.2, 1112.0, 524.9, 494.7,
       79.18, 0, −142.6, 0, 24.37, −92.26, 736.4, 0, 0, 0, 0, −42.71,
      136.9, 5256.0, −262.3, −174.5, −46.8, 77.55, −185.3, 125.3, 4288.0,
      0, 0, 0, 0, 37.1, 0.0, 0, 32.9, 0, −48.33, 0, 336.25, 0, −195.1, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0}, {−72.88, 41.38, −101.12, 614.52, 68.95, 967.71, 0, 0, 443.6,
       −55.87, −111.45, 0, 173.77, 0, 0, 0, 0, 0, 329.1, 0, 0, 0, 0, 0, 0,
      174.4, 0, 0, 0, 0, 0, 0, 0, 0.0, 0, 0, 2073.0, 0, −119.8, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {50.49, 64.07, −2.504,
      −143.2, −25.87, 695.0, −240.0, 0, 110.4, 0, 41.57, 0, −93.51,
      −366.51, 0, −257.2, 0, 0, 0, −180.2, 0, −215.0, −343.6, −58.43,
      −334.12, 0, 0, 0, 85.7, 0, 535.8, 0, −111.2, 0, 0.0, 0, 0, 0, −97.71,
       0, 153.7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {−165.9,
      573.0, −123.6, 397.4, 389.3, 218.8, 386.6, 0, 114.55,
 354.0, 175.5, 0, 0, 0, 0, 0, 0, 0, −42.31, 0, 0, 0, 0, −85.15, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0.0, −208.8, 0, −8.804, 0, 423.4, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {47.41, 124.2, 395.8, 419.1,
      738.9, 528.0, 0, 0, −40.9, 183.8, 611.3, 134.5, −217.9, 0, 0, 0, 0,
      281.6, 335.2, 898.2, 383.2, 301.9, −149.8, −134.2, 0, 379.4, 0,
      167.9, 0, 82.64, 0, 0, 322.42, 631.5, 0, 837.2, 0.0, 0, 255.0, 0,
      730.8, 0, 0, −262.0, 0, 0, 0, 2429.0, 0, 0, 0, 0, −127.06, 0, 0, 0},
      {−5.132000000000001,
 −131.7, −237.2, −157.3, 649.7, 645.9, 0, 0, 0, 0, 0, 0, 167.1, 0, −198.8,
       116.5, 0, 159.8, 0, 0, 0, 0, 0, −124.6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 215.2, −110.65, −117.2, 0, 0, 0, 26.35, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 117.59}, {−31.95, 249.0, −133.9, −240.2, 64.16,
      172.2, −287.1, 0, 97.04, 13.89, −82.12, −116.7, −158.2, 49.7, 10.03,
      −185.2, 343.7, 0, 150.6, −97.77, −55.21, 397.24, 0, −186.7, −374.16,
      223.6, 0, 0, −71.0, 0, −191.7, 0, −176.26, 6.699, 136.6, 5.15,
      −137.7, 50.06, 0.0,
  −5.579, 72.31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 39.84}, {147.3,
      62.4, 140.6, 839.83, 0, 0, 0, 0, 0, 0, 0, 0, 278.15, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 33.95, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 185.6,
      55.8, 0.0, 0, 0, 0, 0, −218.9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
      {529.0, 1397.0, 317.6, 615.8, 88.63, 171.0, 284.4, −167.3, 123.4,
      577.5, −234.9, 65.37, −247.8, 0, 284.5, 0, −22.1, 0, −61.6, 1179.0,
      182.2, 305.4, −193.0, 335.7, 1107.0, −124.7, 0, 885.5, 0, −64.28,
      −264.3, 288.1, 627.7, 0, −29.34, −53.91, −198.0, 0, −28.65, 0, 0.0,
      0, 0, 0, 0, 0, −353.5, 0, 0, 0, 0, 0, 0, −100.53, 0, 0},
  {−34.36, 0, 787.9, 191.6, 1913.0, 0, 180.2, 0, 992.4, 0, 0, 0, 448.5,
      961.8, 1464.0, 0, 0, 0, 0, 2450.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 169.3, 233.1, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0}, {110.2, 0, 234.4, 221.8, 84.85, 0, 0, 0, 0, 0, 0, 0,
      0, −125.2, 1604.0, 0, 0, 0, 0, 2496.0, 0, 0, 0, 70.81, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 745.3, −2166.0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0}, {13.89, −16.11, −23.88, 6.2139999999999995,
      796.9, 0, 832.2, −234.7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      −196.2, 0, 161.5, 0, 0, 0, −274.1, 0, 262.0, 0,
  0, 0, 0, 0, −66.31, 185.6, 0, 0, 0, 0, 0, 26.35, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0}, {30.74, 0, 167.9, 0, 794.4, 762.7, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 844.0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, −32.17, 0, 0, 0, 0, 111.8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0}, {27.97, 9.755, 0, 0, 394.8, 0, −509.3, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, −70.25, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, −322.3, 0, 0, 0, 0, 0, 0, 0, 0, 0},
      {−11.92, 132.4, −86.88, −19.45, 517.5, 0, −205.7, 0, 156.4, 0,
      −3.444, 0, 0, 0, 0, 0, 0, 0, 119.2, 0, 0, −194.7, 0,
      3.1630000000000003, 7.082000000000001, 0, 0, 0, 0, 0, 515.8, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 101.2, 0, 0,
  0, 0, 0, 0.0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {39.93, 543.6, 0, 0, 0, 420.0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, −363.1, −11.3, 0, 0,
      0, 0, 6.971, 0, 0, 0, 0, 0, 0, 0, 148.9, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0.0, 0, 0, 0, 0, 0, 0, 0, 0}, {−23.61, 161.1, 142.9, 274.1, −61.2,
       −89.24, −384.3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0.0, 0, 0, 0, 0, 0, 0, 0, 0}, {−8.479, 0, 23.93, 2.845, 682.5, 597.8,
       0, 810.5,
  278.8, 0, 0, 0, 0, 0, 0, 0, 0, 221.4, 0, 0, 0, 0, 0, −79.34, 0, 176.3, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {245.21, 384.45,
      47.05, 347.13, 72.19, 265.75, 627.39, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 75.04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {21.49, −2.8,
      344.42, 510.32, 244.67, 163.76, 833.21, 0, 569.18, −1.25, −38.4,
      69.7, −375.6, 0, 0, 0, 0, 0, 0, 600.78, 291.1, 0, −286.26, −52.93, 0,
       0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 177.12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0}, {272.82, 569.71, 165.18, 369.89, 0, 0, 0, 0, −62.02, 0,
       −229.01, 0, −196.59, 0, 0, 0, 0, 100.25, 0, 472.04, 0, 0, 0, 196.73,
       0, 0, 0, 434.32, 0, 0, 0, 313.14, 0, 0, 0, 0, 0, 0, 0, 0, −244.59,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 920.49, 305.77,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 171.94, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0}, {−20.31, 0, −106.7, 568.47, 284.28, 0,
      401.2, 0, 106.21, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, −108.37,
      5.76, 0, −272.01, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 107.84, −33.93, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};
    //BIP overview
    // a1,1 a1,2 a1,3 ...  a1,56
    // a2,1
    // a3,1
    // .
    // .
    // .
    // a56,1
    // Data are devloped with the help of python script and DWSIM
    parameter Integer i,j;
    algorithm
    for i in 1:length loop
      for j in 1:length loop
       A[i,j]:=BIP[Main_ID[ID[i]],Main_ID[ID[j]]];
      end for;
    end for;

    for i in 1:length loop
        tau[i, :] := exp((−A[i, :]) / T);
      end for;

  end UNIFAC_BIP;
```

UNIFAC_RQ function (6th in figure 4.1)

```
function UNIFAC_RQ
input Integer NOC;
input Integer length;
input Integer ID_sub[NOC,5,2];
output Real ri[NOC];
output Real qi[NOC];
output Real ei[length,NOC];
protected
//RQ data base
/*[[1, 'CH2', 'CH3'], [2, 'CH2', 'CH2'], [3, 'CH2', 'CH'], [4, 'CH2', 'C
    '], [5, 'C=C', 'CH2=CH'], [6, 'C=C', 'CH=CH'], [7, 'C=C', 'CH2=C'],
    [8, 'C=C', 'CH=C'], [9, 'C=C', 'C=C'], [10, 'ACH', 'ACH'], [11, 'ACH
    ', 'AC'], [12, 'ACCH2', 'ACCH3'], [13, 'ACCH2', 'ACCH2'], [14, 'ACCH2
    ', 'ACCH'], [15, 'OH', 'OH'], [16, 'MeOH', 'CH3OH'], [17, 'H2O', 'H2O
    '], [18, 'ACOH', 'ACOH'], [19, 'CH2CO', 'CH3CO'], [20, 'CH2CO', '
    CH2CO'], [21, 'CHO', 'CHO'], [22, 'COOC', 'CH3COO'], [23, 'COOC', '
    CH2COO'], [24, 'HCOO', 'HCOO'], [25, 'CH2O', 'CH3O'], [26, 'CH2O', '
    CH2O'], [27, 'CH2O', 'CHO'], [28, 'CH2O', 'THF'], [29, 'CNH2', '
    CH3NH2'], [30, 'CNH2', 'CH2NH2'], [31, 'CNH2', 'CHNH2'], [32, 'CNH',
    'CH3NH'], [33, 'CNH', 'CH2NH'], [34, 'CNH', 'CHNH'], [35, 'C3N', '
    CH3N'], [36, 'C3N', 'CH2N'], [37, 'ACNH2', 'ACNH2'], [38, 'Pyridine',
     'C5H5N'], [39, 'Pyridine', 'C5H4N'], [40, 'Pyridine', 'C5H3N'], [41,
     'CCN', 'CH3CN'], [42, 'CCN', 'CH2CN'], [43, 'COOH', 'COOH'], [44, '
    COOH', 'HCOOH'], [45, 'CCl', 'CH2Cl'], [46, 'CCl', 'CHCl'], [47, 'CCl
    ', 'CCl'],
[48, 'CCl2', 'CH2Cl2'], [49, 'CCl2', 'CHCl2'], [50, 'CCl2', 'CCl2'], [51,
     'CCl3', 'CHCl3'], [52, 'CCl3', 'CCl3'], [53, 'CCl4', 'CCl4'], [54, '
    ACCl', 'ACCl'], [55, 'CNO2', 'CH3NO2'], [56, 'CNO2', 'CH2NO2'], [57,
    'CNO2', 'CHNO2'], [58, 'ACNO2', 'ACNO2'], [59, 'CS2', 'CS2'], [60, '
    CH3SH', 'CH3SH'], [61, 'CH3SH', 'CH2SH'], [62, 'Furfural', 'Furfural
    '], [63, 'DOH', 'DOH'], [64, 'I', 'I'], [65, 'Br', 'Br'], [66, 'C#C',
     'CH#C'], [67, 'C#C', 'C#C'], [68, 'DMSO', 'DMSO'], [69, 'ACRY', '
    Acrylonitrile'], [70, 'ClCC', 'Cl-C=C'], [71, 'ACF', 'ACF'], [72, '
    DMF', 'DMF'], [73, 'DMF', 'HCON(CH2)2'], [74, 'CF2', 'CF3'], [75, '
    CF2', 'CF2'], [76, 'CF2', 'CF'], [77, 'COO', 'COO'], [78, 'SiH2', '
    SiH3'], [79, 'SiH2', 'SiH2'], [80, 'SiH2', 'SiH'], [81, 'SiH2', 'Si
    '], [82, 'SiO', 'SiH2O'], [83, 'SiO', 'SiHO'], [84, 'SiO', 'SiO'],
    [85, 'NMP', 'NMP'], [86, 'CClF', 'CCl3F'], [87, 'CClF', 'CCl2F'],
    [88, 'CClF', 'HCCl2F'], [89, 'CClF', 'HCClF'], [90, 'CClF', 'CClF2'],
     [91, 'CClF', 'HCClF2'], [92, 'CClF', 'CClF3'], [93, 'CClF', 'CCl2F2
    '], [94, 'CON', 'CONH2'], [95, 'CON', 'CONHCH3'], [96, 'CON', '
    CONHCH2'], [97, 'CON', 'CON(CH3)2'], [98, 'CON', 'CONCH3CH2'], [99, '
    CON', 'CON(CH2)2'], [100, 'OCCOH', 'C2H5O2'], [101, 'OCCOH', 'C2H4O2
    '], [102, 'CH2S', 'CH3S'], [103, 'CH2S', 'CH2S'], [104, 'CH2S', 'CHS
    '], [105, 'Morpholine', 'morpholine'], [106, 'Thiophene', 'C4H4S'],
    [107, 'Thiophene', 'C4H3S'], [108, 'Thiophene', 'C4H2S'], [109, '
    CH2SuCH2', 'CH2SuCH2'], [110, 'CH2SuCH2', 'CH2SuCH'], [111, 'Oxides',
     'CH2OCH2'], [112, 'Oxides', 'CH2OCH'], [113, 'Oxides', 'CH2OC'],
    [114, 'Oxides', 'CHOCH'], [115, 'Oxides', 'CHOC'], [116, 'Oxides', '
    COC'], [117, 'Anhydride', 'O=COC=O'], [118, 'Aromatic Nitrile', '
    AC-CN'], [119, 'Aromatic Bromo', 'AC-Br']]*/
parameter Real RQ[119,2] = {{0.9011, 0.848}, {0.6744, 0.54}, {0.4469,
    0.228}, {0.2195, 0.0}, {1.3454, 1.176}, {1.1167, 0.867}, {1.1173,
    0.988}, {0.8886, 0.6759999999999999}, {0.6605, 0.485}, {0.5313, 0.4},
     {0.3652, 0.12}, {1.2663, 0.968}, {1.0396, 0.66}, {0.8121, 0.348},
    {1.0, 1.2}, {1.4311, 1.432}, {0.92, 1.4}, {0.8952, 0.68}, {1.6724,
    1.4480000000000002}, {1.4457, 1.18}, {0.998, 0.948}, {1.9031,
    1.7280000000000002}, {1.6764, 1.42}, {1.242, 1.188}, {1.145, 1.088},
    {0.9183, 0.78}, {0.6908, 0.46799999999999997}, {0.9183, 1.1},
    {1.5959, 1.544}, {1.3692, 1.236}, {1.1417, 0.924}, {1.4337, 1.244},
    {1.207, 0.9359999999999999}, {0.9795, 0.624}, {1.1865, 0.94},
    {0.9597, 0.632}, {1.06, 0.816}, {2.9993, 2.113}, {2.8332,
    1.8330000000000002}, {2.667, 1.5530000000000002}, {1.8701, 1.724},
    {1.6434, 1.416}, {1.3013, 1.224}, {1.528, 1.5319999999999998},
    {1.4654, 1.264}, {1.238, 0.9520000000000001}, {1.0106,
    0.7240000000000001}, {2.2564, 1.9980000000000002}, {2.0606,
    1.6840000000000002}, {1.8016, 1.4480000000000002}, {2.87, 2.41},
    {2.6401, 2.184}, {3.39, 2.91}, {1.1562, 0.8440000000000001}, {2.0086,
1.868}, {1.7818, 1.56}, {1.5544, 1.248}, {1.4199, 1.104}, {2.057, 1.65},
    {1.8769999999999998, 1.676}, {1.651, 1.368}, {3.168, 2.484}, {2.4088,
     2.248}, {1.264, 0.992}, {0.9492, 0.8320000000000001},
    {1.2919999999999998, 1.088}, {1.0613, 0.784}, {2.8266, 2.472},
    {2.3144, 2.052}, {0.7909999999999999, 0.7240000000000001}, {0.6948,
    0.524}, {3.0856, 2.736}, {2.6322, 2.12}, {1.406, 1.38}, {1.0105,
    0.92}, {0.615, 0.46}, {1.38, 1.2}, {1.6035, 1.263}, {1.4443, 1.006},
    {1.2853, 0.7490000000000001}, {1.047, 0.41}, {1.4838, 1.062},
    {1.3030000000000002, 0.764}, {1.1044, 0.466}, {3.9810000000000003,
    3.2}, {3.0356, 2.6439999999999997}, {2.2287, 1.916}, {2.406, 2.116},
    {1.6493, 1.416}, {1.8174, 1.6480000000000001}, {1.9669999999999999,
    1.828}, {2.1721, 2.1}, {2.6243, 2.376}, {1.4515, 1.248}, {2.1905,
    1.796}, {1.9637, 1.4880000000000002}, {2.8589, 2.428}, {2.6322,
    2.12}, {2.4054, 1.8119999999999998}, {2.1226, 1.9040000000000001},
    {1.8952, 1.5919999999999999}, {1.6130000000000002, 1.368}, {1.3863,
    1.06}, {1.1589, 0.748}, {3.4739999999999998, 2.7960000000000003},
    {2.8569, 2.14}, {2.6908, 1.86}, {2.5247, 1.58}, {2.6869, 2.12},
    {2.4595, 1.808}, {1.5926, 1.32}, {1.3652, 1.008}, {1.1378, 0.78},
    {1.1378, 0.696}, {0.9103, 0.46799999999999997}, {0.6829, 0.24},
    {1.7732, 1.52}, {1.3342, 0.996}, {1.3629, 0.972}};
//Read the value RQ[RQ_ID,:];  return [Rk, Qk] array
Integer i ,j, k;
algorithm
qi:=zeros(NOC);
ri:=zeros(NOC);
ei:=zeros(5,NOC);
//surface area constant
  for i in 1:NOC loop
    for j in 1:length loop
    k :=ID_sub[i,j,1];
      if k>0 then
        qi[i] := qi[i] + ID_sub[i,j,2] .* RQ[k,2];
        ri[i] := ri[i]+ ID_sub[i,j,2] .* RQ[k,1];
      else
        qi[i] := qi[i]+0;
        ri[i] := ri[i]+0;
      end if;
    end for;
  end for;

  for i in 1:NOC loop
    for j in 1:length loop
    k :=ID_sub[i,j,1];
      if k>0 then
        ei[j, i] := ID_sub[i,j,2] .* RQ[k,2] / qi[i];
      else
        ei[j, i] := 0;
      end if;
    end for;
  end for;
end UNIFAC_RQ;
```

This flow is fully automated but Temperature is the input variable of the function loop so there is high probability of error in flash which does not contain the temperature is input variable.

There is also a second possible flow which is semi automated; user have to find the length of the Unique sub ID array form the length algorithm first and then manually add it to the unifac. Then they have to call the BIP function only for the A matrix and RQ function for V, R, Q array. This flow is very use full when we implement our python GUI for system, python first calculate the length then input it to the model.

Part III

Bug Fixing in 1.13.2

Chapter 5

ShortCut Column

5.1 Convergence problem

For find out the root of problem, I run provided test simulation of shortcut column and use the debugger to understand the nature of problem.
Test simulator provided with the OMChemSim: Error is created by the intermediate variable (Here it is $cse13) which is directly connected with the bottoms.T.
Same test with Peng-Robinson: error in all Thermo parameter and mole flow and composition.
New developed Water-Ethanol simulation with Raoults Law and NRTL: error in Distillate.T equation.

5.2 Possible error inside the ShortCut column

Hence the error is created at the distillate.T or bottoms.T; It is possible that error are in the cond.T or reb.T because this both are connected with the distillate.T and bottoms.T.

5.2.1 Confirmation of this error

As per debugger error is in bottom.T or distillate.T but instantiate models show that this strings are connected with reb.T or cond.T and the other intermediate are connected with the comp[i].VP and VP function are used ShortCut column. So we can tell that error may be lie inside the ShortCut column model.

5.2.2 Solution of that error

Detail analysis of the debugger tell that log(rebT) and log(distT) are the root of the problem so, I just use A=log(rebT) and B=log(distT) and provide them a boundary condition;

old code

if condType == "Partial" then

```
  1/condP = sum(mixMolFrac[3, :]./ (gamma[:] .* exp(comp[:].VP[2] + comp[:].VP[3] /
  condT + comp[:].VP[4] * log(condT) + comp[:].VP[5] .* condT .^ comp[:].VP[6])));
  rebP = sum(gamma[:] .* mixMolFrac[2, :] .* exp(comp[:].VP[2] + comp[:].VP[3] /
  rebT + comp[:].VP[4] * log(rebT) + comp[:].VP[5] .* rebT .^ comp[:].VP[6]));
```

elseif condType == “Total” then

```
  condP = sum(gamma[:] .* mixMolFrac[3, :] .* exp(comp[:].VP[2] + comp[:].VP[3] /
  condT + comp[:].VP[4] * log(condT) + comp[:].VP[5] .* condT .^ comp[:].VP[6]));

  rebP = sum(gamma[:] .* mixMolFrac[2, :] .* exp(comp[:].VP[2] + comp[:].VP[3] /
  rebT + comp[:].VP[4] * log(rebT) + comp[:].VP[5] .* rebT .^ comp[:].VP[6]));
```

end if;

Code after applying condition

```
if rebT<=0 and condT<=0 then
  log(rebT)=1;
  log(condT)=1;
    ..... old bunch of code
else if rebT<=0 then
  log(rebT)=1;
    ..... old bunch of code
else if condT<=0 then
  log(condT)=1;
    ..... old bunch of code
else
    ..... old bunch of code
end if;
```

After this solution Flowsheet are converged but there is error in minR (value of minR=0).
But from this solution we can tell that problem’s root is in this logarithmic values.

5.3 minR<=0 error

5.3.1 Error due to theta

Theta are defined for calculation of Underwood equation which has same root as NOC; but most of the time root is taken as zero or negative.
Possible solution of this error is to find the positive root, or use the alternative method for Underwood equation.
For that we try to use the Modelica.Math.Nonlinear.solveOneNonlinearEquation but this is not able to satisfy all condition we require.

Underwood Equation

$$vapPhasMolFrac = \sum_{i=1}^{NOC} \frac{\alpha[i] * mixMolFrac[1, : i]}{\alpha[i] - theta}$$

$$minR + 1 = \sum_{i=1}^{NOC} \frac{\alpha[i] * mixMolFrac[3,:i]}{\alpha[i] - theta}$$

Here to find the perfect value of the **theta** we break first equation into the smaller parts; dummyA and dummyB.

$$dummyA = \alpha[i] * mixMolFrac[1,:i]$$

$$dummyB = \alpha[i] - theta$$

As par literature theta must be positive and $\alpha[HKey] < theta < \alpha[LKey]$.
To generate the exception for other condition we use other if ... ... else loop for it. And this condition is applicable to the dummyB; So, we modify the theta in dummyB with dummyTheta to provide the nessesary condition. So now $dummyB = \alpha[i] - dummyTheta$.

New code with the condition

```
if dummyTheta<0 then
  theta=0;
  root =dummyTheta -theta;
elseif dummyTheta==0 then
  theta=0;
  root =dummyTheta-theta;
elseif dummyTheta > alpha[LKey] then
  root = dummyTheta - theta;
  theta = dummyTheta;
elseif dummyTheta < alpha[HKey] then
  root = dummyTheta -theta;
  theta = dummyTheta;
else
  dummyTheta = theta;
  root = 1;
end if;
```

Idea behind this Exception development is very simple. Here, after every condition we save the dummyTheta's value in the theta and here theta work as intermediate value storage. If the dummyTheta's vlaue is 0 or negative or not in between $\alpha[HKey] < theta < \alpha[LKey]$ then we multiply first equation's right side with the $root = dummyTheta - theta$ because if the theta (which store the value of dummyTheta) are one zeros of equation then it is divisible by root and by doing that we can eliminate that value, When the iteration scheme are enter into the selected condition root become 1 and equation does not affected by the condition and after several iteration we gor perfect value of theta.

After this two modification our shortcut column is versatile and work for multicompound also (in previous version it only handle 3 compound but after this modification it also work with 5 compound).
I also add one other veriable "verify" to verify that exception does not fail, if the verify is non zero then there is error in the "theta" and it is not the root of first equation of underwood.

5.3.2 error in distillate vapor fraction and Reflux ratio

To tackle with this problem we further brake 2^{nd} underwood's equation into two parts, dummyC and dummyD.

$$dummyC[i] = \alpha[i] * mixMolFrac[3, i]$$
$$dummyD[i] = \alpha[i] - theta$$

By braking equation we simplify it for the solver for handling the iterative procedure to find the unknown variable.

5.3.3 Improvement in solution

but that equation set is also written as

```
if dummyTheta > alpha[LKey] or dummyTheta < alpha[HKey] then
  theta = dummyTheta;
  root = dummyTheta - theta;
else
  dummyTheta = theta;
  root = 1;
end if;
```

by observing this equations we can tell that root become zero when condition is not satisfying. SO we write this as

```
if theta > alpha[LKey] or theta < alpha[HKey] then
  root = 0;
else
  root = 1;
end if;
```

and we remove the dummyTheta from equation and it works! so after that we remove all other dummy variable and again check the value and Shortcut work perfectly.

```
if theta > alpha[LKey] or theta < alpha[HKey] then
  0= sum((alpha[:] .* mixMolFrac[1, :])./ (alpha[:] .- theta));
  //This is mathamatical adjustment for right convergence of theta
else
  vapPhasMolFrac[1] = sum((alpha[:] .* mixMolFrac[1, :])./ (alpha[:] .- theta));
end if;
```

And this is also represent the same situation!

Bibliography

[1] Byung Ik Lee and Michael G. Kesler; “A Generalized Thermodynamic Correlation Based on Three-Parameter Corresponding States”; AlChE Journal (Vol. 21, No. 3), Page 510 May, 1975

[2] Ulf Plócker, Helmut Knapp, and John Prausnltz; “Calculation of High-Pressure Vapor Liquid Equilibria from a Corresponding-States Correlation with Emphasis on Asymmetric Mixtures”; Ind. Eng. Chem. Process Des. Dev., Vol. 17, No. 3, 1978