```python
#When F1=700

get_ipython().run_line_magic('reset', '-f')

import numpy as np
import pandas
import sympy
from termcolor import colored


###### inputs

stream_direction = ["in","in","out","out"] # directions of streams
component_labels = ["Flow rate","Methene","Ethene","Propyne","Nitrogen","Oxygen","CO2"]
fraction_prefix = "y" # mass (x) or mole (y) fraction

###### process relation

process_equation = ["F1=0.25*F2"] ## eg:"F1=0.25*F2"

###### input fraction , flow rate

#F1=700
y1M=0.33
y1E=0.29
y2N=0.79
y3M=0.1
y3P=0.3
y4E=0.06
y4O=0.07

y1N=0; y1O=0; y2M=0; y2E=0; y2P=0; y2C=0; y3N=0; y3O=0; y4P=0; y4C=0;

######
```

→透過 list 隨意輸入 process relation

```
Ns = len(stream_direction)
Nc = len(component_labels) - 1
Np = len(process_equation)
```

→ process_equation的長度就是Np數量

```


stream_labels = []
for i in range(Ns):
    stream_labels.append("stream " + str(i+1) + "-" + stream_direction[i])


V_empty = np.empty([Ns, Nc+1],dtype=object)


df = pandas.DataFrame(V_empty, index=stream_labels, columns=component_labels)
print (df)


all_variables = []


for i in range(len(stream_direction)):
    all_variables.append(component_labels[0][0] + str(i+1))


for i in range(len(component_labels[1:])):
    for j in range(len(stream_direction)):
        all_variables.append(fraction_prefix + str(j+1)
                             + component_labels[i+1][0])



all_symbols = sympy.symbols(all_variables)
```

```
V_initial = V_empty.copy()

given_variables = []

unknown_index=[]
for i in range(len(all_variables)):


    index_row = int(all_variables[i][1]) - 1

    if all_variables[i][0] == component_labels[0][0]:
        index_column = 0

    else:
        for j in range(len(component_labels)):

            if all_variables[i][2] == component_labels[j][0]:
                index_column = j

    if all_variables[i] in globals():

        V_initial[index_row, index_column] = eval(all_variables[i])
        given_variables.append(all_variables[i])
    else:
        V_initial[index_row, index_column] = all_symbols[i]
        unknown_index.append([index_row,index_column])


print("\n")


df = pandas.DataFrame(V_initial, index=stream_labels, columns=component_labels)
print (df)
```

```
N_zeros = np.count_nonzero(V_initial == 0)
Nv = Ns*(Nc+1) - N_zeros + Np
Nd = Nv - Ns - Nc -Np
```

⇒ 加入Np後的Nv, Nd

```
print ("\nThe number of total variables (zeros not included):", Nv)
print (colored("The number of design variables:",attrs=["bold"]), Nd)
print ("\nYou have given", len(given_variables)-N_zeros+Np, "variables")
```

```
if Np != 0:
    print("\nThere are",Np,"process relations")
    for i in process_equation:
        print(i)
```

⇒ Np≠0時 print出 process relation的數量
並將process relation print出

```
# Check if you have given too many or didn't give enough design variables
if len(given_variables)-N_zeros+Np > Nd:
    print (colored("You have given too many input variables","red"))
if len(given_variables)-N_zeros+Np < Nd:
    print (colored("You didn't give enough input variables","red"))
```

⇒ 加入Np後的判斷式

```
u=0
for i in range(len(stream_direction)):
    if all_variables[i] not in globals():
        u+=1
    if u==len(stream_direction):
        print(colored("\nYou didn't provide any Flow information so can't solve the problem","red"))
```

如果沒有定義任何Flow rate, 即顯示無法解

```
140 ##########################################################################################################
141
142 sym_equations=[]
143
144 if Np!= 0:
145     function=process_equation
146     for i in range(len(process_equation)):
147         temp_eq=[]
148         n=0
149         for j in range(len(process_equation[i])):
150             if process_equation[i][j] == "=":
151                 n=1
152             if n>0:
153                 temp_eq.append("-("+process_equation[i][j+1:]+")")
154                 break
155             if n==0:
156                 temp_eq.append(process_equation[i][j])
157         function[i]="".join(temp_eq)
158
159 for i in range(len(process_equation)):
160     sym_equations.append(sympy.Eq((sympy.sympify(function[i])),0))
161
162
163 ##########################################################################################################
164
165 unknown_variables = [element for element in all_variables if element not in given_variables]
166 number_unknowns = len(unknown_variables)
167 print ("\nThere are", number_unknowns, "unknown variables:")
168 print (*unknown_variables,sep=" ,")
169
170 signed_V = V_initial.copy()
171
172 for i in range(signed_V.shape[0]):
173     if stream_direction[i] == "out":
174         signed_V[i,0] = signed_V[i,0]*-1
```

當Np≠0時

整理process equation

先尋找process equation中等號的位置

並省略等號將等號右邊所有的東西

括號起來並加上負號

最後放入sym_equation中

```
functions = []

for i in range(Nc):
    functions.append(sum(signed_V[:,0]*signed_V[:,i+1]))

for i in range(Ns):
    functions.append(sum(signed_V[i,1:]) - 1)


for i in functions:
    sym_equations.append(sympy.Eq(i,0))


solutions = sympy.solve(sym_equations,unknown_variables)

solutions = eval(str(solutions))


Final_V = V_initial.copy()

if len(solutions)!=0:

    print("\nThe solutions are:")

    for i in range(len(unknown_variables)):
        print(unknown_variables[i],"=",sympy.Float(solutions[0][i],5))

    for i in range(len(unknown_variables)):
        Final_V[unknown_index[i][0],unknown_index[i][1]] =sympy.Float(solutions[0][i],5)

    for i in range(len(solutions[0])):
        if int(solutions[0][i])<0:
            print(colored("\nWarning !! The solution contains negative numbers","red"))
            break
```

如果process_equation中有已知項,最終的答案會得到解析解,所以還要將已知代入答案

如果solution中有負數將提出警告

```
210
211 elif len(solutions)==0 and len(given_variables)-N_zeros+Np == Nd:
212
213     print(colored("\nWarning there should be existing dependent equations","red",attrs=["bold"]))
214
215 print ("\n")
216
217
218 df = pandas.DataFrame(Final_V, index=stream_labels, columns=component_labels)
219 df
220
```

如果可以解但 solution 為空集合代表有 dependent equation 存在