トルエンの空気酸化による安息香酸の製造

1講座 移動現象論分野

荊尾太雅 , 宮本奏汰

keyword

空気酸化	Air oxidation
気液反応	Gas-liquid reaction
晶析	Crystallization
多変数同時全体最適化	Multi-variable simultaneous optimization
空気	air

目次

第 1 章

緒言

安息香酸は，主としてフェノールの原料となる他，その塩が食品や化粧品などの添加物として広く利用されている．2014 年には世界全体で 48 万トンが製造されており，新興国での需要の増加から，2024 年には生産量が 64 万トンとなることが見込まれている []．そこで，安価なトルエンを原料として用いて安息香酸を製造するプロセスを検討することにした．

第 2 章

プロセスの概要

2.1　プロセスの概要

本設計で対象とするのは，トルエンを空気酸化することにより安息香酸を製造するプロセスである．プロセス全体の概略図を図 2.1 に示す．

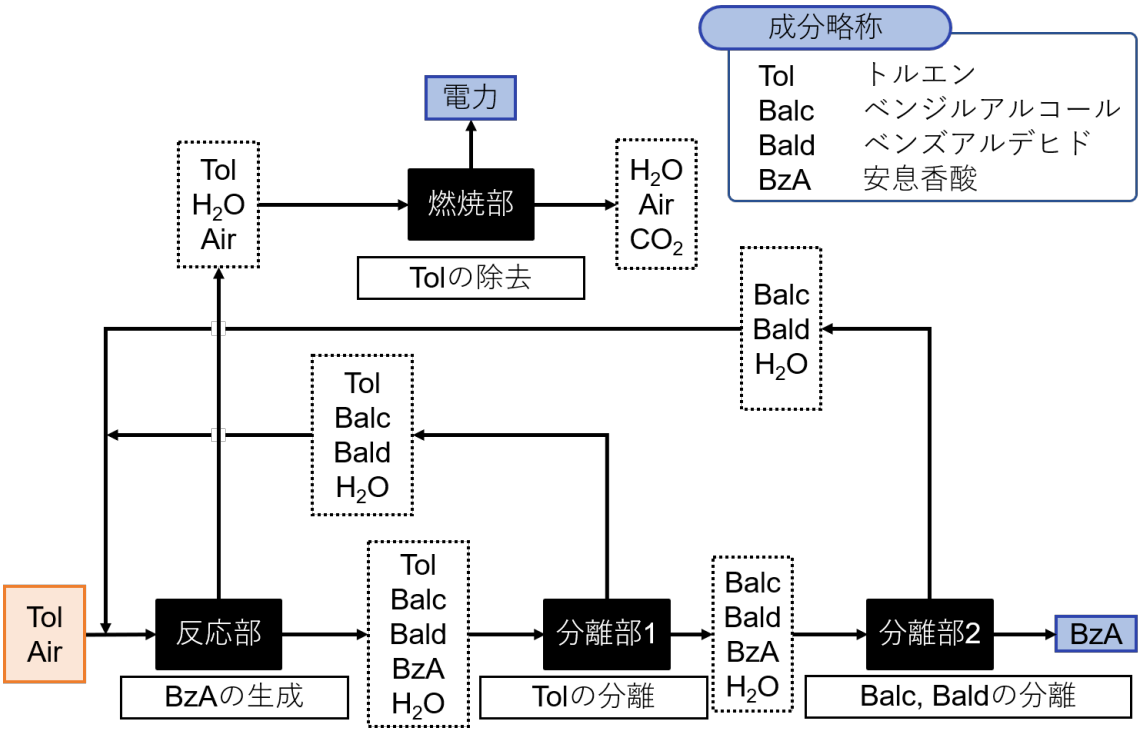

図 2.1　プロセス全体の概略図

2.2　設計条件

プロセスを設計するにあたり，以下の条件を設定した．

1. 生産要求は，99.0 wt% 以上の安息香酸を年 2 万トンとする．

2. 工場の稼働時間は，1 日 24 時間，年 300 日とする．

3. 原料として，純度 100 % のトルエンおよび，組成を窒素 79 mol% 酸素 21 mol% とする空気を用いる．ただし，両原料は 25 degreeCelsius，1 bar で供給されるものとする．

4. 減価償却期間は 8 年とする．

5. 圧力損失，熱損失，制御系については考慮しない．

6. HYSIS を用いての物性推算は UNIQUAC 式によって行う．

第 3 章

反応部

トルエンを空気酸化させ，安息香酸に転化させることを目的とする工程である．反応部の概略図を図 3.1 に示す．反応器にフィードされる液は，原料およびリサイクルによって回収された未反応トルエン，ベンジルアルコール，ベンズアルデヒド，水であり，反応器入り口において 7 bar，170 °C に加熱および加圧され，反応器に供給される．また，反応器底部からは空気が 1 bar，25 °C で供給されており，攪拌されながら反応器上方に向かう．反応器内は外部熱媒によって加熱され，7 bar，170 °C に保たれており，攪拌によって液中に溶けた酸素と未反応物質は触媒酸化反応を起こす．また，気液間物質移動により，液中のトルエンおよび水が盛んに蒸発するが，蒸発したトルエンおよび水を含む空気は反応器からコンデンサーに流入し凝縮される．凝縮したトルエンおよび水はデカンターへ送られ，密度差分離によりトルエンのみを反応器に還流し，水はパージする．また，凝縮しなかったトルエンは，環境安全上の理由からそのまま排出することはできないので，濃度を減少させるために燃焼部へと送られる．反応器からの流出液は次の工程である分離部 1 へと送られる．

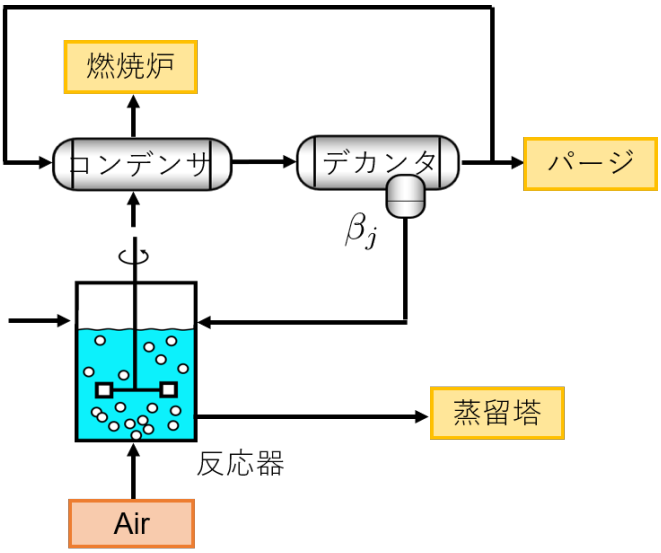

図 3.1　反応部概略図

3.1　反応機構

今回用いた反応を以下に示す．それぞれの反応速度式および，298 K における反応エンタルピーも示している．トルエンは経路によらず最終的に安息香酸へと転化する．

$$
\begin{array}{llll}
\mathrm{Tol} + \mathrm{O_2} & \longrightarrow \mathrm{Bald} + \mathrm{H_2O} & r_1 = k_1 \cdot C_{\mathrm{Tol}} & \Delta_{\mathrm{r}}H_1 = -385.23 \ \mathrm{kJ\,mol^{-1}} \\
\mathrm{Tol} + 1/2\ \mathrm{O_2} & \longrightarrow \mathrm{Balc} & r_2 = k_2 \cdot C_{\mathrm{Tol}} & \Delta_{\mathrm{r}}H_2 = -167.30 \ \mathrm{kJ\,mol^{-1}} \\
\mathrm{Balc} + 1/2\ \mathrm{O_2} & \longrightarrow \mathrm{Bald} + \mathrm{H_2O} & r_4 = k_3 \cdot C_{\mathrm{Balc}} & \Delta_{\mathrm{r}}H_3 = -317.30 \ \mathrm{kJ\,mol^{-1}} \\
\mathrm{Bald} + 1/2\ \mathrm{O_2} & \longrightarrow \mathrm{BzA} & r_3 = k_4 \cdot C_{\mathrm{Bald}} & \Delta_{\mathrm{r}}H_4 = -298.20 \ \mathrm{kJ\,mol^{-1}}
\end{array}
$$

ただし，トルエンを Tol，ベンジルアルコールを Balc，ベンズアルデヒドを Bald，安息香酸を BzA と表記している．これらの反応は全て酸化反応であり，触媒には，酸化剤であるモリブデン酸マンガン $\mathrm{MnMoO_4}$ を用いることとした．これらの反応の反応速度定数は，表 3.1 のようである []．

表 3.1　各反応式の反応速度定数

	k_1	k_2	k_3	k_4
$\ln A_j$ [−]	17.93	20.63	15.40	19.70
E_j [kJ mol^{-1}]	69.53	81.39	56.99	71.44

また，酸化反応が激しい場合には燃焼反応によって二酸化炭素まで転化するので，反応器における反応率が高い場合には考慮する必要があるが，今回の設計結果では単通反応率が 0.265 と低かったため，燃焼反応は無視できると考えた．

3.2　反応器選定

流通式の気液反応器の種類には，拡散速度に対して不利な順に，気泡塔，気液攪拌槽，充填塔などがある．反応器を選定するにあたり，最大反応速度と最大拡散速度の比を表す八田数を概算し，八田数が 0.1 より小さいなら気泡塔，5 より大きければ充填塔，中間域ならば気液攪拌槽を選択することとした [1]．今回は気液攪拌槽型反応器を選択した．実際のプロセスでは気泡塔も選択される．

反応器内攪拌には 6 枚羽根タービン翼を用い，空気流入による冷却作用が大きいため，ジャケットを取り付け，ジャケット内部に熱媒を流している．また，空気を反応器内に送るスパージャー直径は反応器直径の 1/3 とした．

八田数の定義式

$$
\gamma = \frac{(\text{最大反応速度})}{(\text{最大拡散速度})} = \frac{(C_{\mathrm{B}} k D_{\mathrm{B}})^{1/2}}{k_{\mathrm{L}}} \tag{3.1}
$$

3.3　設計方程式

両相の滞留時間の大きさが十分異なると考えられることから，液相は完全混合状態として，気相は鉛直方向に向かう押し出し流れと仮定できると判断した．設計時に用いた仮定は以下のようである．

- 気相は水平方向に一様な濃度分布を持つ．
- 気相側境膜抵抗は無視できる
- 液相は完全混合状態である．
- 窒素，酸素はヘンリー則に従い，その他の物質はラウール則に従う．

以上の仮定および，蒸発油分を還流する機構を含めて，設計方程式 (3.2)，(3.3)，(3.4) を立式した．

$$
0 = F^{\mathrm{in}}_{\mathrm{liq},j} - F^{\mathrm{out}}_{\mathrm{liq},j} - (1-\beta_j) k_{\mathrm{L}} a \int_0^{V_{\mathrm{tot}}} (C_j - C_j^{\mathrm{sat}}) \mathrm{d}V + r_j V_{\mathrm{L}} \tag{3.2}
$$

$$\frac{\mathrm{d}F_{\mathrm{gas},j}}{\mathrm{d}V} = k_{\mathrm{L}}a(C_j - C_j^{\mathrm{sat}}) \tag{3.3}$$

$$\sum_j F_{j,\mathrm{in}}H_{j,\mathrm{in}} - \sum_j F_{j,\mathrm{out}}H_{j,\mathrm{out}} = UA(T_{\mathrm{s}} - T) \tag{3.4}$$

解析方法としては，まず油分を全還流として近似的に各液相中濃度を決定した．そして，トルエンおよび，酸素，窒素について濃度を仮定した後に，気相の物質収支式を Runge-kutta 法によって計算し，各蒸発量が収支式と一致する濃度を求めた．また，物質移動容量係数が十分大きいと判断し，最適化中の解析に用いる場合には，気相は迅速に平衡状態へ達すると仮定した．

3.4 物質移動容量係数の推算

物質移動容量係数は，反応装置形状，反応器内部流体の様々な物性，流れの状態に依存する. 攪拌槽型反応器に対し，物質移動容量係数の推算に用いた各相関式を記す．

液相側物質移動係数の相関式 [1]

$$\text{小気泡の場合：} \quad k_{\mathrm{L}} = 0.31Sc_{\mathrm{L}}^{-2/3}(g\Delta\rho\mu_{\mathrm{L}}/\rho_{\mathrm{L}}^2)^{1/3} \tag{3.5}$$

$$\text{大気泡の場合：} \quad k_{\mathrm{L}} = 0.42Sc_{\mathrm{L}}^{-1/2}(g\Delta\rho\mu_{\mathrm{L}}/\rho_{\mathrm{L}}^2)^{1/3} \tag{3.6}$$

比表面積の相関式 [1]

$$a = 1.44\left(\frac{P_{\mathrm{V}}^{0.4}\rho_{\mathrm{L}}^{0.2}}{\sigma^{0.6}}\right)\left(\frac{u_{\mathrm{G}}}{u_{\mathrm{t}}}\right)^{0.5}\left(\frac{P_{\mathrm{T}}}{P_{\mathrm{G}}}\right)\left(\frac{\rho_{\mathrm{G}}}{\rho_{\mathrm{a}}}\right)^{0.16} \tag{3.7}$$

上記の 2 式を利用するために用いた相関式を以下に記す．

ガスホールドアップの相関式 [1]

$$\varepsilon_{\mathrm{G}} = \left(\frac{u_{\mathrm{G}}\varepsilon_{\mathrm{G}}}{u_{\mathrm{t}}}\right) + 0.000216 \times \left(\frac{P_{\mathrm{V}}^{0.4}\rho_{\mathrm{L}}^{0.2}}{\sigma^{0.6}}\right)\left(\frac{u_{\mathrm{G}}}{u_{\mathrm{t}}}\right)^{0.5}\left(\frac{P_{\mathrm{T}}}{P_{\mathrm{G}}}\right)\left(\frac{\rho_{\mathrm{G}}}{\rho_{\mathrm{a}}}\right)^{0.16} \tag{3.8}$$

気泡の体積平均径の相関式 [1]

$$d_{\mathrm{vs}} = 4.15\left(\frac{\sigma^{0.6}}{P_{\mathrm{V}}^{0.4}\rho_{\mathrm{L}}^{0.2}}\right)\left(\frac{P_{\mathrm{G}}}{P_{\mathrm{T}}}\right)\left(\frac{\rho_{\mathrm{a}}}{\rho_{\mathrm{G}}}\right)^{0.16}\varepsilon_{\mathrm{G}}^{0.5} + 0.0009 \tag{3.9}$$

気泡の終末速度の相関式 [1]

$$u_{\mathrm{t}} = \left(\frac{4\Delta\rho g d_{\mathrm{vs}}}{3C_{\mathrm{D}}\rho_{\mathrm{L}}}\right)^{0.5} \tag{3.10}$$

さらに，上記の相関式を利用するために用いた物性値の推算式，および変数の定義式などの諸式を以下に記す．

抗力係数の相関式 [1]

$$C_{\mathrm{D}} = \max\left[\frac{24}{Re}(1 + 0.15Re^{0.687}),\ \frac{8}{3}\frac{Eo}{Eo+4}\right] \tag{3.11}$$

拡散係数の推算

wilke-chang の式 [1]

$$D_{12} = \frac{2.946 \times 10^{-11}(\beta M_{\mathrm{r},2})^{1/2}T}{\mu_2 V_{\mathrm{b},1}^{0.6}} \tag{3.12}$$

Einstin-Stokes の式 [8]

$$\frac{D\mu}{T} = \mathrm{const} \tag{3.13}$$

界面張力の推算
界面張力の温度依存性に関する相関式 [1]

$$\sigma \propto \{1 - (T/T_{\mathrm{c}})\}^{n} \tag{3.14}$$

気泡レイノルズ数の定義式 [1]

$$Re = \frac{\rho_{\mathrm{L}} u_{\mathrm{t}} d_{\mathrm{vs}}}{\mu_{\mathrm{L}}} \tag{3.15}$$

エトベス数の定義式 [1]

$$Eo = \frac{g \Delta\rho d_{\mathrm{vs}}^{2}}{\sigma} \tag{3.16}$$

また，参考文献 [1] からトルエンの標準沸点における分子容積を $\mathrm{cm^3\,mol^{-1}}$ とし，参考文献 [10] からトルエンの界面張力 $\sigma = 0.81\,\mathrm{N\,m^{-2}}$ とし，参考文献 [9] から会合度を 1 とした．

以上の諸式を用いて物質移動容量係数を推算した．結果は反応器設計結果で示す．

3.5 反応器設計結果

最終結果における反応器の詳細を表 3.2 に示す．ただし，反応器内温度は，外部熱媒を用いることで，170 °C に保たれているとした．

表 3.2 反応器設計結果

項目	値
反応器内圧力 [bar]	7.00
反応器内温度 [°C]	170
反応器体積 [$\mathrm{m^3}$]	14.6
気液総体積 [$\mathrm{m^3}$]	8.25
単通販能率 [–]	0.265
ガスホールドアップ [–]	0.115
液平均滞留時間 [h]	0.478
物質移動容量係数 [$\mathrm{s^{-1}}$]	0.470
液相物質移動係数 [$\mathrm{m\,s^{-1}}$]	0.00172
比界面積 [$\mathrm{m^{-1}}$]	273
反応器空間率 [–]	0.435
総攪拌動力 [kW]	8.25
気泡体積平均径 [mm]	2.53
エトベス数 [–]	1480
気泡レイノルズ数 [–]	1170

3.6 反応部設計結果

最適化後の最終結果における反応部の物質収支を図 3.2 中に示す．最適化方法については最適化の章で述べる．

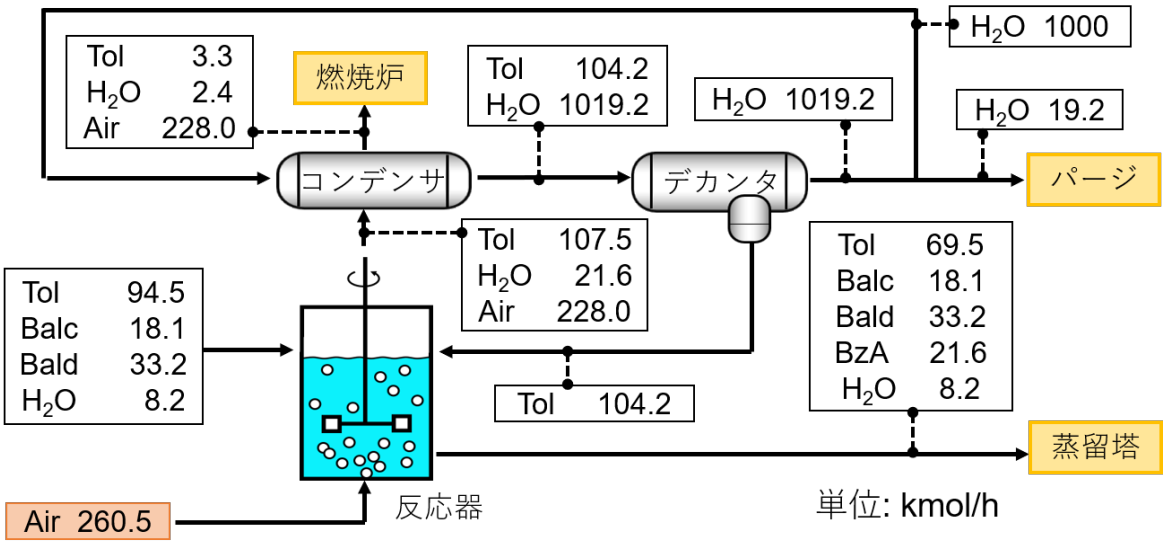
Tol 3.3
H2O 2.4
Air 228.0
燃焼炉
Tol 104.2
H2O 1019.2
H2O 1019.2
H2O 1000
H2O 19.2
コンデンサ
デカンタ
パージ
Tol 107.5
H2O 21.6
Air 228.0
Tol 94.5
Balc 18.1
Bald 33.2
H2O 8.2
Tol 104.2
Tol 69.5
Balc 18.1
Bald 33.2
BzA 21.6
H2O 8.2
蒸留塔
Air 260.5
反応器
単位: kmol/h

図 3.2　反応部設計結果

第 4 章

分離部 1

4.1　蒸留塔設計

未反応トルエンのうち 99 % 以上を回収することを目的とした．設計条件として，蒸留塔段数を 10 段，蒸留塔供給段は 6 段，還流比を 1.0 とした．

蒸留塔圧力を変更し，全体の利益を最大とする点について探索を行った．反応器流出液圧力が 7 bar であるため，最適な圧力であることが予想される．蒸留塔の圧力を変化させ，人件費を除く全体の利益を評価関数としてプロットした図 4.1 から，確かに 7 bar で全体の利益が最大化されることを確認した．

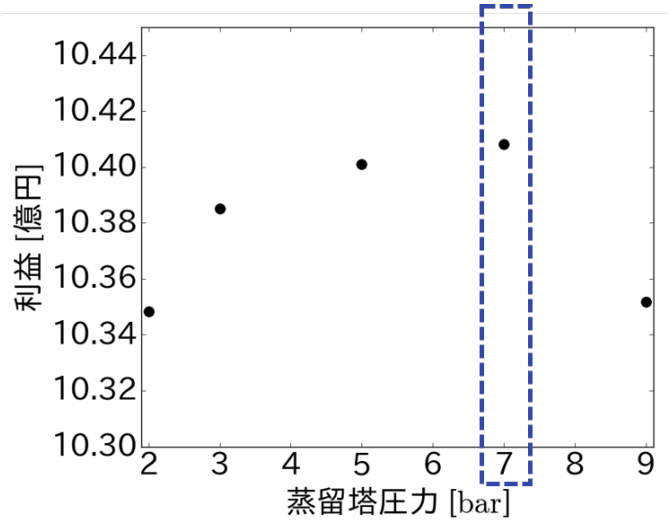

図 4.1　蒸留塔圧力最適化

設計結果を表 4.1 に記す．

表 4.1　蒸留塔設計結果

項目	値
塔径　[m]	1.0
塔高　[m]	6.1
塔内圧力　[bar]	7
コンデンサ内温度　[°C]	232
リボイラー内温度　[°C]	316

4.2　分離部 1 設計結果

最適化後の最終結果における分離部 1 の流量関係を図 4.2 中に示す.

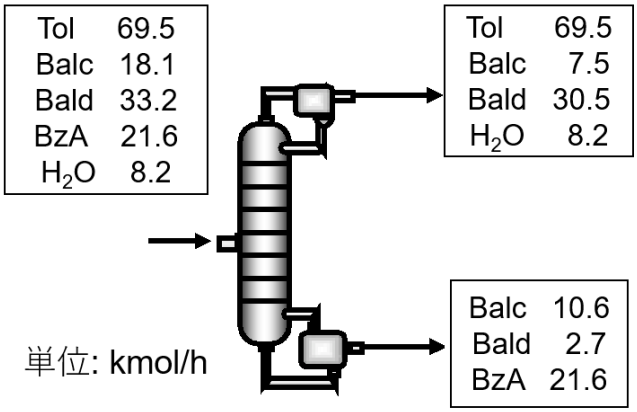

図 4.2　分離部 1 設計結果

第 5 章

分離部 2

5.1　晶析器選定

溶解度の温度依存性が大きいことと，目的とする結晶生産量が大きいことから，連続式攪拌槽型反応装置を選定した．

溶解度の温度依存性の相関式

$$C^* = 2.03 \times 10^{-5} T^4 + 2.03^{-5} \times T^4 + 2.97 \times 10^{-4} T^3 + 4.70 \times 10^{-2} T^2 + 1.43T + 24.71 \tag{5.1}$$

ただし, 温度 T [°C]，溶解度 C^* [g kg-solvent^{-1}]

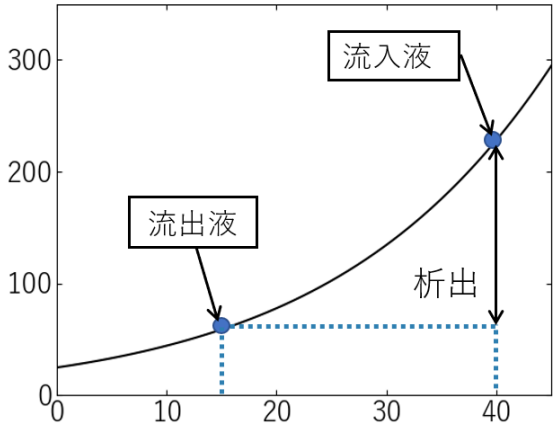

図 5.1　溶解度の温度依存性

5.2　設計方程式

以下の仮定を用いた．

- 結晶表面拡散は迅速に行われる．
- 晶析器内は完全混合状態である．
- 二次核発生の影響は無視する．

参考文献 [2] により，以下の実験式および理論式を用いて設計を行った．

一次核発生速度

$$B^0 = k_\mathrm{b} M_\mathrm{T}^j \Delta C^b \tag{5.2}$$

結晶成長速度

$$G = k_\mathrm{g} \Delta C^g \tag{5.3}$$

結晶成長速度定数

$$k_\mathrm{g} = k_\mathrm{g0} \exp\left(-\frac{E_\mathrm{g}}{RT}\right) \tag{5.4}$$

個数収支式

$$n = n^0 \exp\left(-\frac{L}{G\tau}\right) \tag{5.5}$$

懸濁密度

$$M_\mathrm{T} = c_0 - c = 6 k_\mathrm{v} \rho_\mathrm{c} n^0 (G\tau)^4 \tag{5.6}$$

各パラメータは以下の通りである.

$k_\mathrm{g0} = 1.06 \times 10^7\ (\mu\mathrm{m})(\mathrm{g/g\text{-}solvent})^{\text{-}g}$
$B^0 = 40.05\,\mathrm{kJ\,mol^{-1}}$
$k_\mathrm{b} = 9.16 \times 10^{12}\ (\#/\mathrm{m^3 s})(\mathrm{g/mL})^{\text{-}j}(\mathrm{g/g\text{-}solvent})^{\text{-}b}$
$g = 0.44$
$j = 1.78$
$b = 1.2$
$k_\mathrm{v} = 0.1$
$\rho_\mathrm{c} = 1.32\,\mathrm{g\,cm^{-3}}$

5.3 晶析器設計結果

最適化の結果によって得られた晶析器の設計結果を表 5.1 に記す.

表 5.1 晶析器設計結果

項目	値
晶析器体積 [$\mathrm{m^3}$]	7.16
晶析器液体積 [$\mathrm{m^3}$]	3.58
晶析器フィード液温 [°C]	40.0
晶析器内温度 [°C]	13.3
晶析器内圧力 [bar]	1.00
滞留時間 [min]	9.71
単通結晶収率 [−]	0.690
結晶化可能量基準収率 [−]	0.902
結晶の体積平均径 [μm]	3.41

5.4　抽出塔設計

十分に塔内へ液を滞留させることによって安息香酸を水中に飽和させることを目的とした．十分なデータを得られなかったため，2 時間の装置内滞留によって安息香酸がトルエン溶媒中に飽和し，その他ベンジルアルコール，ベンズアルデヒドの溶解度については無視できると仮定した．

5.5　分離部 2 設計結果

分離部 2 のみの流量関係を簡易的に図 5.2 中に示す．蒸留塔から送られてきた油液は冷却され，抽出塔内に供給される．晶析器で用いた安息香酸水溶液を加熱し，溶解度を上昇させて抽出塔内に供給する．抽出塔内では安息香酸が水中に溶解し，ベンジルアルコール，ベンズアルデヒドおよび触媒は溶解せずに回収され，反応部へ送られる．安息香酸水溶液を晶析器に供給し，製品結晶を得ている．

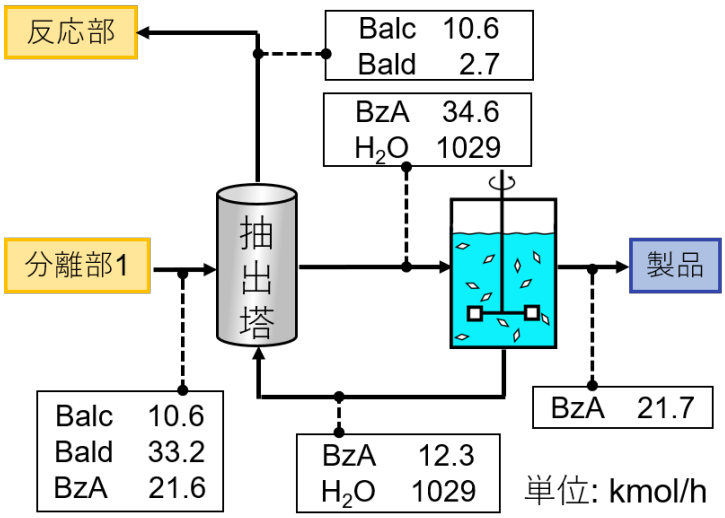

図 5.2　分離部 2 設計結果

第 6 章

燃焼部

トルエンは人体に有害な物質であり，吸引によって神経系などに深刻な被害を与えることもある．東京都の条例には排出基準が厳しく定められており，この基準を満たすため，反応部のコンデンサーで凝縮しなかったトルエンを燃焼させることでその濃度を減じる．燃焼炉内ではトルエンが完全燃焼し，二酸化炭素および水へ転化すると仮定し，量論的に燃焼に必要な量と等量の酸素を供給して燃焼炉へと送るものとした．燃焼熱は熱媒の作成に使用され，足りない分は燃料を投入して賄っているものと考えた．

第 7 章

最適化

7.1　方法

最適化を行うにあたり，最適化変数として以下の 3 つを選択した．

- 反応器体積
- 晶析器体積
- 晶析器温度

これらの変数を選択した理由を述べる．反応器体積が大きいとき，反応器の装置コストは大きくなる．一方で，反応器出口から排出される未反応トルエンの量が減るため，蒸留塔のリボイラーで必要な熱量が減少し，熱煤のコストは減少する．反応器体積が小さいときは，反応器の装置コストは小さくなるが，熱煤のコストは大きくなる．晶析器体積が大きいとき，晶析器の装置コストは大きくなる．しかし，晶析する安息香酸の量が増えるため，リサイクルに回る安息香酸の量が少なくなり，それを保持するための純水の量が減少し，純水の費用は抑えられる．晶析器の体積が小さいときは，逆になる．晶析器温度が低い時，必要な外部冷媒の温度が低くなるため，冷媒の単価は高くなる．しかし，晶析する安息香酸の量が大きくなるため，リサイクルに回る安息香酸の量が小さくなり，必要な純水の量が減るため，純水を冷やすために用いている冷媒の量は小さく抑えられる．晶析器温度が高いときは，冷媒の単価は安くなるが，冷媒の必要量が大きくなる．このように，最適化変数に選択した変数は，その大小によって，なんらかのトレードオフの関係を生じる．そのため，これらの変数の値を変化させることで，利益が最大となる最適点を探索することが可能であると考えた．

次に，最適化の具体的な手法について述べる．プロセスを最適化するにあたり，3 つの最適化変数変数全てを同時に最適化することを考えた．そこで，反応器体積は，$13.5 \sim 16.0\,\mathrm{m}^3$ の範囲で 6 点，晶析器体積は，$6 \sim 12\,\mathrm{m}^3$ の範囲で 13 点，晶析器温度は，$5.0 \sim 20.0\,^\circ\mathrm{C}$ の範囲で 16 点，計 1248 点のデータを取った．そして，それぞれのデータについてフィッティングを行った後に，それぞれの範囲を 100 当分し，データの個数を $100 \times 100 \times 100$ 点に増やした．それらのデータを用いて，横軸に晶析器温度，縦軸に晶析器体積を設定し，反応器体積の値を逐次的に変化させることで，100 枚のヒートマップを作成した．ヒートマップが表す値は，

$$\text{P.I.} = (\text{売上}) - (\text{原料コスト}) - (\text{装置コスト}) - (\text{用役コスト}) \tag{7.1}$$

で表される評価関数の値とした．また，各ヒートマップ上で，評価関数の値が最大となる点をプロットした．さらに，そのプロットがどのように変化するかを見るために，横軸に反応器体積，縦軸に評価関数の値を取り，折れ線グラフを作成した．

7.2　最適化結果

評価関数の値が最大となった点での，ヒートマップおよび，評価関数の最大値の変化を表したグラフは図 7.1 のようになった．また，作成した gif は github にアップしているので，興味のある方はご覧ください（`https://github.com/miyamo-sou/processdesign`）．最適点での反応器体積は 14.69 m^3，晶析器体積は 7.16 m^3，晶析器温度は 13.3 °C となった．また，その時の評価関数の値は 10.35 億円となった．

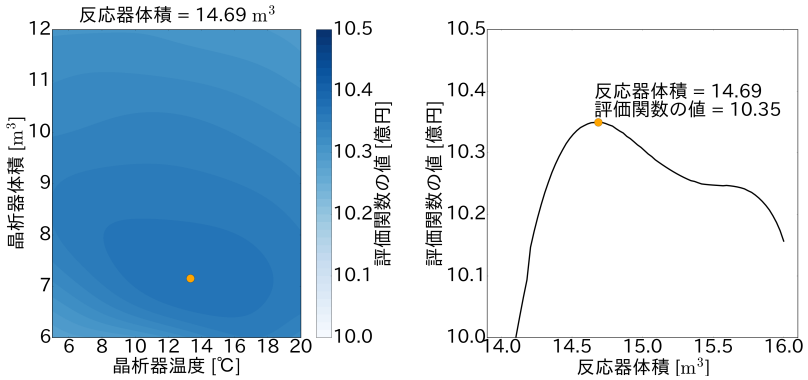

図 7.1　評価関数の値が最大となる点でのヒートマップと最適点の推移

このよに，評価関数は大きなピークが存在する形になっていることが分かる．これは，最適化方法のセクションでも述べたが，反応器体積，晶析器体積，晶析器温度を変化させると，装置費と用役費がトレードオフの関係となって，コストが変化する．そのため，このようにピークが生じたと考えられる．

第 8 章

物質収支・熱収支

全体のフロー図は図 8.1 のようになった.

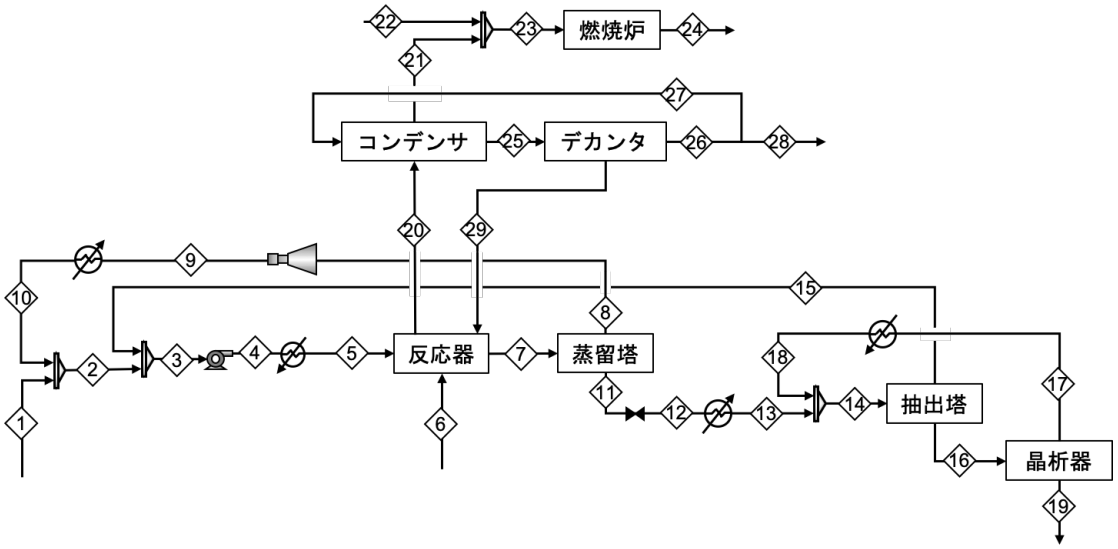

図 8.1　フロー図

また，フロー図に記載されているフローの組成および温度，圧力は表 8.1，8.2，8.3 のようになった.

表 8.1　流量関係

[kmol h^{-1}]	1	2	3	4	5	6	7	8	9	10
トルエン	25	94.5	94.5	94.5	94.5	0	69.5	69.5	69.5	69.5
ベンジルアルコール	0	7.5	18.1	18.1	18.1	0	18.1	7.5	7.5	7.5
ベンズアルデヒド	0	30.5	33.2	33.2	33.2	0	33.2	30.5	30.5	30.5
安息香酸	0	0	0	0	0	0	21.6	0	0	0
水	0	8.2	8.2	8.2	8.2	0	8.2	8.2	8.2	8.2
空気	0	0	0	0	0	260.5	0	0	0	0
二酸化炭素	0	0	0	0	0	0	0	0	0	0
合計	25	140.7	154	154	154	260.5	150.6	115.7	115.7	115.7
温度　[°C]	25	79.9	75.56	76.04	170	25	170	231.5	194.8	90.88
圧力　[bar]	1	1	1	7	7	1	7	7	1	1

表 8.2　流量関係

[kmol h^{-1}]	11	12	13	14	15	16	17	18	19	20
トルエン	0	0	0	0	0	0	0	0	0	107.5
ベンジルアルコール	10.6	10.6	10.6	10.6	10.6	0	0	0	0	0
ベンズアルデヒド	2.7	2.7	2.7	2.7	2.7	0	0	0	0	0
安息香酸	21.6	21.6	21.6	34.6	0	34.6	13	13	21.6	0
水	0	0	0	1029	0	1029	1029	1029	0	21.6
空気	0	0	0	0	0	0	0	0	0	228
二酸化炭素	0	0	0	0	0	0	0	0	0	0
合計	34.9	34.9	34.9	1076.9	13.3	1063.6	1042	1042	21.6	357.1
温度　[°C]	315.6	230.9	25	25	40	40	13.3	40	13.3	25
圧力　[bar]	7	1	1	1	1	1	1	1	1	1

表 8.3　流量関係

[kmol h^{-1}]	21	22	23	24	25	26	27	28
トルエン	3.4	0	3.4	0	104.1	0	0	0
ベンジルアルコール	0	0	0	0	0	0	0	0
ベンズアルデヒド	0	0	0	0	0	0	0	0
安息香酸	0	0	0	0	0	0	0	0
水	2.4	0	2.4	15.8	0	0	0	0
空気	228	39.4	267.4	237.8	1019.2	1019.2	1000	19.2
二酸化炭素	0	0	0	23.4	0	0	0	0
合計	233.8	39.4	273.2	277	1123.3	1019.2	1000	19.2
温度　[°C]	25	25	25	25	25	25	25	25
圧力　[bar]	1	1	1	1	1	1	1	1

フロー全体の与熱流体および，受熱流体は図 8.2 のようになった．図中の流体の熱量関係は表 8.4 および，8.5 のようになった．

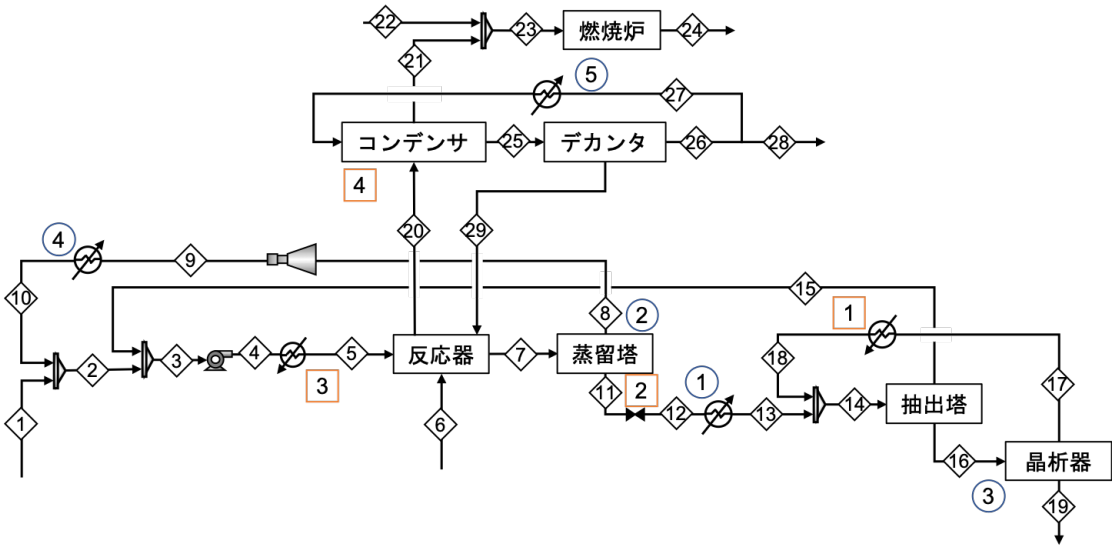

図 8.2　与熱流体と受熱流体

表 8.4　与熱流体

与熱流体	変化前温度 [°C]	変化後温度 [°C]	与熱量 [MJ h^{-1}]
① 蒸留塔留出液	194.8	90.9	6338.4
② 蒸留塔コンデンサー	248.3	231.5	5111.8
③ 晶析器流入流体	40.0	13.3	2252.7
④ 蒸留塔缶出液	230.9	25	2436.7
⑤ デカンタ流出純水	40.0	25	1130.6

表 8.5　受熱流体

受熱流体	変化前温度 [°C]	変化後温度 [°C]	受熱量 [MJ h^{-1}]
[1] 晶析器リサイクル	13.3	40.0	2149.3
[2] 蒸留塔リボイラー	304.0	315.6	11735.8
[3] 反応器入り口流体	76.0	170.0	2922.4
[4] デカンタ流入純水	25.0	40.0	1130.6

第 9 章

ヒートインテグレーション

流体同士の熱交換を行い，外部流体の利用量を削減することを目的としてヒートインテグレーションを行った．本プロセスにおいては熱交換器は多管型熱交換器として，向流で熱交換を行った．熱交換面積を求めるため，以下の式を用いた．

$$Q = UA(\Delta T)_{\mathrm{lm}} \tag{9.1}$$

ただし $(\Delta T)_{\mathrm{lm}}$ は温度差の対数平均であり，熱交換によって与熱流体の温度が T_{h1} から T_{h2} に変化し，受熱流体の温度が T_{c2} から T_{c1} に変化するとき，

$$(\Delta T)_{\mathrm{lm}} = \frac{(T_{\mathrm{h1}} - T_{\mathrm{c1}}) - (T_{\mathrm{h2}} - T_{\mathrm{c2}})}{\ln\{(T_{\mathrm{h1}} - T_{\mathrm{c1}})/(T_{\mathrm{h2}} - T_{\mathrm{c2}})\}} \tag{9.2}$$

と表される．総括熱伝達係数として，両熱交換流体の相状態にのみ依存するとして表 9.1 の値を用いた．

表 9.1　総括熱伝達係数

流体 1	流体 2	総括伝熱係数 [$\mathrm{W\,m^{-1}\,s^{-1}}$]
ガス	ガス	150
ガス	液	200
ガス	ガス (凝縮)	500
ガス	液 (蒸発)	500
液	液	300
液	ガス (凝縮)	1000
液	液 (蒸発)	1000
ガス (凝縮)	液 (蒸発)	1500

以下に，用いた外部熱媒の温度と熱量を表**??**に示す．

図 9.1 に最終的な設計結果における TQ 線図を示す．

ヒートインテグレーションを行うことで，必要な外部熱媒の費用は 3.6 億円から 1.2 億円にまで削減することができた．

表 9.2　用いた外部熱媒

熱媒	温度 [°C]	交換熱量 [$\mathrm{kJ\,h^{-1}}$]
熱媒 1	330.0	11735.8
熱媒 2	55.0	1085.7
熱媒 3	2.3	1761.1

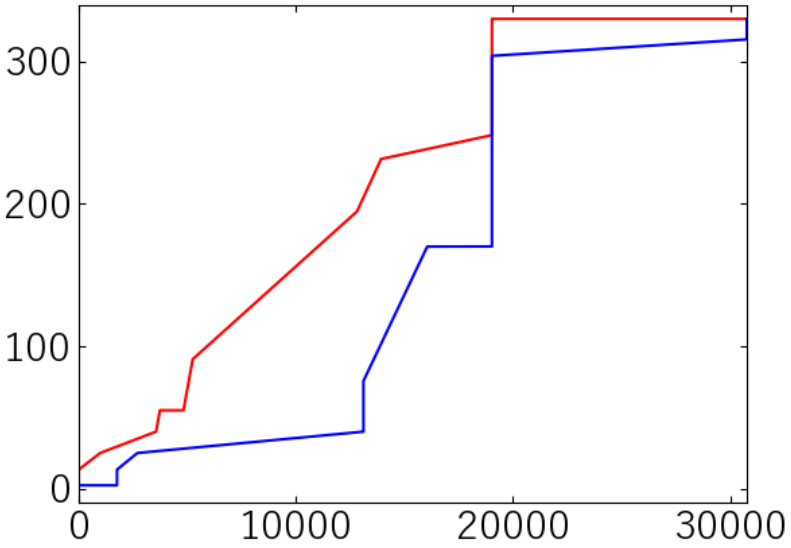

図 9.1　TQ 線図

第 10 章

経済評価

減価償却を 8 年とした場合の経済評価は表 10.1 のようになった．なお，各装置費の推算式は，Appendix A に示した通りである．また，トルエンは 500 $/t，安息香酸は 1013 $/t とした．

表 10.1　経済評価 [億円/年]

収入	製品	安息香酸	23.7	23.7
支出	原料コスト	トルエン	10.4	10.4
	装置コスト	晶析器	0.39	18.31
		反応器	0.34	
		熱交換器	0.33	
		抽出塔	0.14	
		コンデンサ	0.05	
		燃焼炉	0.05	
		蒸留塔	0.04	
		デカンタ	0.04	
		ポンプ	0.02	
		コンプレッサー	0.02	
	運転コスト	用役コスト	1.24	6.60
		触媒	0.36	
		人件費	5	

第 11 章

結言

設計目標を純度 99.0 wt% の安息香酸を年 2 万 ton 製造するものとして，設計を行った．文献値を参考として気液反応器，および晶析装置を設計した．リサイクルフローを考えて，原料を有効に利用すル事ができた．プロセスを最適化するにあたり，反応器体積，晶析器体積，晶析器内温度の 3 変数を同時に最適化することができた．結果として，より精度のよい最適化を行うことが可能となった．これにより，年間 5.39 億円の利益を見込める設計を行うことができた．

残った課題としては，蒸発トルエンのさらなる回収を行うため，燃焼炉ではなく吸着装置を用いることを検討することや，各装置についてさらに詳細に設計することが挙げられる．

謝辞

今回のプロセス設計では，様々な方々にお世話になりました．山本教授，谷口准教授をはじめとする多くの化学工学の先生方，集中講義を実施してくださった玉川先生に感謝の意を申し上げます．

また，1 講座の先輩方の協力無しには私たちのプロセス設計は実現できませんでした．プロセスの内容や発表に対し，的確なアドバイスをいただきました．お忙しい中，原稿やスライドのチェック，発表練習などに協力して頂いたことで，無事に発表を終えることができました．

私たちのプロセス設計に協力してくださった先生方，先輩方に改めて御礼申し上げます．

参考文献

[1] 冨山明男, 片岡勲, 坂口忠司. 気泡の抗力係数に関する研究 : 第 1 報, 静止液中単一気泡の抗力係数. 日本機械学会論文集 B 編, Vol. 61, No. 587, pp. 2357–2364, 1995.

[2] Wang Li, Qingjun Zhang, and Aiwu Zeng. Kinetics and mechanism modeling of liquid-phase toluene oxidation to benzaldehyde catalyzed by mn–mo oxide. *Transactions of Tianjin University*, Vol. 25, No. 1, pp. 52–65, Feb 2019.

参考文献

[1] 化学工学会・化学工学便覧・丸善出版

[2] Gary Morris and Graham Power *et al. Org. Process Res. Dev.* **19**, 1891-1902, 2015

[3] 富士フィルム和光純薬 (株) https://labchem-wako.fujifilm.com/jp/product/detail/W01W0113-0065.html

[4] キシダ化学 (株) http://www.kishida.co.jp/product/catalog/detail/id/960

[5] 化学工学会・SIS 部会・情報技術教育分科会, 第 10 回プロセスデザイン学生コンテスト

[6] 京都大学・プロセス設計講義資料 3

[7] 第 6 回ソフトウェアツール学生コンテスト http://www.chemeng.titech.ac.jp/~sis_cont/dairokkai.html

[8] 京都大学化学プロセス工学コース 実験テキスト

[9] R.Byron Bird and Warren E. Stewart ,Edwin N. Lightfoot TransPortPhenomena revised second edition willy

[10]

変数一覧

a:比界面積
B^0:一次核発生速度
c:重量濃度
C:モル濃度
C_{D}:抗力係数
D:拡散係数
d_{vs}:気泡体積平均径
E:活性化エネルギー
E_{g}:結晶化過程の活性化エネルギー
F:モル流量
g:重力加速度
G:核成長速度
k:反応速度定数
k_{b}:核発生速度定数
k_{g}:核成長速度定数
k_{L}:液相物質移動係数
$k_{\mathrm{L}}a$:液相物質移動容量係数
k_{v}:結晶体積形状係数
M_{T}:懸濁密度
n:結晶の個数密度
P_{G}:攪拌動力
P_{T}:
r:反応速度
T:温度
T_{c}:臨界温度
u_{t}:気泡の終末速度
u_{G}:気泡の空塔速度
V:体積
β:還流率
μ:粘度
μ_{L}:液相粘度
σ:界面張力
ρ_{a}:空気密度
ρ_{L}:液相密度
ρ_{g}:気相密度
$\Delta\rho$:密度差
Eo:エトベス数
Sc:シュミット数
Re:レイノルズ数

付録 A

コスト推算

為替レートは 1 ドル=111.73 円とした．(2019 年 4 月平均)

A.1　労務費

労務費の推算について補足する．プラントは 4 直 3 交代で運転され，1 班の人数に関して次の推算式を用いた．

$$(1\text{ 班の人数}) = (6.29 + 0.23 \times (\text{主要機器数}))^{0.5} \tag{A.1}$$

よって，総運転員数は 40 人であり，主任などに 10 人加え，50 人に平均 1000 万円の給与を支払うとして労務費を算出した．

A.2　ユーティリティコスト

用役単価を表 A.1 に示す．触媒であるモリブデン酸マンガンは参考文献 [2] によると，材料である酢酸

表 A.1　用役単価

項目	価格
燃料 [$/ GJ]	1.095
触媒 [円/g]	9.616
2.3 ℃プロピレン冷媒 [$/ GJ]	4.804
純水 [$/ t]	45
電力 [$/ kW h]	0.1

マンガン 21 g とモリブデン酸アンモニウム 5.04 g を混ぜて作っている．各価格について酢酸マンガンは 500 g - 3100 円とし ([3]，2019/7/17 確認)，モリブデン酸マンガンは 500 g - 11900 円 ([4]，2019/7/17 確認) とした．燃料，プロピレン冷媒 (内挿値) の価格は参考文献 [5] より得た．電力の価格は参考文献 [6] から得た．

さらに [6] から，排水処理費用としてデカンターからパージする純水に 1 t あたり 0.041 $かかるとした．

A.3　機器コスト (参考 [6])

下記の推算は 2001 年のデータで行い，2001 年のコストインデックス 394 と 2018 年のコストインデックス 603.1 を用いて値を修正している．

主要機器について，

1) 常圧で運転することを想定し，炭素鋼を用いて作成されるとしてメーカー出荷地点での価格を推算
2) 関連する部分の直接費，間接費を含めた価格の推算
3) 圧力や材質利用に関する補正

という 3 段階によって機器の建設費を推定する方法を用いた．

まず，メーカー船積み出荷価格 C_{p}^{0} は，機器の特徴サイズ A と係数 K_1, K_2, K_3 を用いて

$$\log_{10} C_{\mathrm{p}}^{0} = K_1 + K_2 \log_{10} A + K_3 (\log_{10} A)^2 \tag{A.2}$$

直接費や間接費，特殊材料費，操作圧力を考慮すると，各装置に関係するコストは C_{p}^{0} の数倍になる．すなわち，

$$C_{\mathrm{BM}} = F_{\mathrm{BM}} C_{\mathrm{p}}^{0} \tag{A.3}$$

と表される．C_{BM}, F_{BM} はそれぞれベアモジュールコスト，ベアモジュールファクターと呼ばれる．F_{BM} は，

$$F_{\mathrm{BM}} = B_1 + B_2 F_{\mathrm{p}} F_{\mathrm{M}} \tag{A.4}$$

と表される．ここで，B_1, B_2, F_{p}, F_{M} はそれぞれ圧力，材質に依存しない部分の係数，依存する部分の係数，圧力ファクター，材質ファクターを表す．F_{p} に関しては槽型の装置に対して推算式 (A.5) が提示されている．

$$F_{\mathrm{p,vessel}} = \begin{cases} \max\left\{ \dfrac{(P_{\mathrm{g}}+1)D}{10.71 - 0.00756(P_{\mathrm{g}}+1)} + 0.5\,,\, 1 \right\} & (P_{\mathrm{g}} > -0.5\,\mathrm{bar}) \\ 1.25 & (P_{\mathrm{g}} \le -0.5\,\mathrm{bar}) \end{cases} \tag{A.5}$$

ただし，P_{g} はゲージ圧 [bar] である．槽型以外の装置については，次式を用いた．

$$\log_{10} F_{\mathrm{p}} = C_1 + C_2 \log_{10} P_{\mathrm{g}} + C_3 (\log_{10} P_{\mathrm{g}})^2 \tag{A.6}$$

各機器のコスト算出に当たって用いた係数を表 A.2 に示す．データが資料にない機器についてはベアモジュールファクターを 1 とした．

表 A.2　ベアモジュールファクター算出に用いた係数

	A	K_1	K_2	K_3	B_1	B_2	C_1	C_2	C_3	材質	F_{M}
反応器	体積 [m^3]	4.5587	0.2986	0.002	1.49	1.52	-	-	-	Ti clad	4.8
晶析器	体積 [m^3]	4.5097	0.1781	0.1344	1.49	1.52	-	-	-	Ti alloy clad	9.4
蒸留塔 (槽)	体積 [m^3]	3.4974	0.4485	0.1074	1.49	1.52	-	-	-	Ti clad	4.8
蒸留塔 (トレイ)	面積 [m^2]	2.9949	0.4465	0.3961	1.49	1.52	-	-	-	Ti clad	4.8
抽出塔	体積 [m^3]	3.4974	0.4485	0.1074	1.49	1.52	-	-	-	Ti	9.4
デカンター	体積 [m^3]	3.4974	0.4485	0.1074	2.25	1.82	-	-	-	CC	1
燃焼炉	燃焼熱量 [kW]	3.068	0.6597	0.0194	-	-	0	0	0	CC	1
ポンプ	電力 [kW]	3.8696	0.3161	0.122	1.89	1.35	0	0	0	Ni alloy	3.9
コンプレッサー	電力 [kW]	2.2897	1.3604	-0.103	-	-	0	0	0	CC	1

熱交換器については，伝熱面積を $A\,[\mathrm{m}^2]$ として次の計算式を用いた [7]．

$$(\text{コスト [億円]}) = 0.015K \times A^{0.65} \tag{A.7}$$

ただし，K は定数でありコンデンサーは 1，リボイラーは 2 とした．

付録 B

プログラム

B.1 python

ソースコード B.1 気液反応器の解析

```
#%%


#気液反応器の解析をします.特に断りがなければSI 単位です
#FlowNO; 0=反応器への流入液,1=反応器からの流出液,2=反応器への流入空気,
# 3=反応からデカンターへの蒸気
# 4=反応器への還流油分,5=デカンターからの流出空気,
# 6=デカンターに入れる冷却水,7=デカンターからの流出水
#ComponentNO; 0=Tol,1=Bol,2=Bal,3=Bac,4=H2O,5=N2,6=O2

import math as mt
import numpy as np
import matplotlib.pyplot as plt
import openpyxl as px




#------------------外部入力値-----------------------------------------------------
#結果出力用ファイル設定（ソースファイルディレクトリ）
conclusionbookname = "conclusion.xlsx"

# 0. 外部入力条件を受けとります
#液相体積
VL = 0.020
#流入液体積 [m3/s]
v0 = 0.020/3600.
#反応器体積
DHrate = 1.0
#全圧
Prea = 7.0e+5
#温度
```

```
Trea = 150. + 273.15
#流入液モル分率
x0 = np.array([0.9, 0., 0., 0., 0.1, 0., 0.])
#流入液全モル量 [mol/s]
F0all = 0.005


#-------------------------------------------------------------------------------
# 1 入力
# 1-1 計算に用いる物性パラメータを入力します
#自然定数
pi = mt.pi
Rgas = 8.314
gra = 9.81

#アレニウス型反応速度式のパラメータ
#頻度因子 [s-1]
A = np.array([mt.exp(20.634), mt.exp(17.928), mt.exp(15.4), mt.exp(19.698)])
A = A / 3600.
#活性化エネルギー
E = np.array([81.389, 69.53, 56.987, 71.442])
E = E * 1000.
#反応速度定数 [s-1]
k = np.array([A[i] * mt.exp(-E[i] / (Rgas * Trea)) for i in range(4)])


# 物性値（HYSIS 150℃）
rhoL = 738.6
muL = 0.1781 / 1000.
rhoG = 5.74

#物性値（推算）
#気相から液相への拡散係数 wilke-chang の式とアインシュタイン-ストークスの式によって計算
DL = 0.0001 * 0.000000074 * ((1. * 92.141) ** 0.5 * 293.15)
      / (0.579 * (29.9 ** 0.6))
DL = (Trea / 293.15) * ((0.579 / 1000.) / muL) * DL


#表面張力 温度依存性の表式でデータを換算
sig = 0.0221 * ((591.7 - Trea) / (591.7 - 353.15)) ** 1.24


# 1-2 反応器の装置条件を決めます
#液相部高さ #反応器直径,断面積
Drea = (VL / (pi * DHrate)) ** (1 / 3)
hL = Drea * DHrate
Area = pi * (Drea ** 2)
#ノズル直径,断面積
```

```
Dnoz = Drea / 1.5
Anoz = pi * (Dnoz ** 2)


# 1-3 反応器の運転条件を決めます
#空間時間
tau = VL / v0
#体積当たり撹拌動力
Pv = 2000.
#反応器への空気体積流量
QG = (50. / 1000. / 3600.) * (VL / 0.00025) ** 0.8
#デカンター内温度
Tdec = 50. + 273.15
#気体の空塔速度
uG = QG / Area
#気体の投入線速度
u = QG / Anoz
#撹拌動力
PG = Pv * VL
#気体の流入運動エネルギー
PK = 0.5 * QG * rhoG * u ** 2
#気体の流入位置エネルギー
Pg = QG * rhoL * gra * hL
#総エネルギー
PT = PG + PK + Pg


# 1-3 マテリアルフローに必要な変数を入力します

T2 = 20. + 273.15

#各フローのモル分率
#x0
x1 = np.array(np.zeros(7))
y2 = np.array([0, 0, 0, 0, 0, 0.79, 0.21])
y3 = np.array(np.zeros(7))
y4 = np.array([1., 0, 0, 0, 0, 0, 0])
y5 = np.array([0., 0., 0., 0., 0., 0., 0.])
y6 = np.array([0., 0., 0., 0., 1., 0., 0.])
y7 = np.array([0, 0, 0, 0, 1., 0, 0])


#各フローのモル流量
F0 = np.array(x0 * F0all)
F1 = np.array([0.] * 7)
F2 = np.array([0., 0., 0., 0., 0., ((Prea * 0.79) / (Rgas * T2)), \
    ((Prea * 0.21) / (Rgas * T2))])
F2 = np.array(F2 * QG)
```

```
F3 = np.array([0.] * 7)
F4 = np.array([0.] * 7)
F5 = np.array([0.] * 7)
F6 = np.array([0.] * 7)
F7 = np.array([0.] * 7)

#各フローのモル濃度
C0 = np.array(F0 / v0)
C1 = np.array([0.] * 7)

#-------------------------------------------------------------------------------
# 2 kLa の推算を行います


#体積平均気泡径と終末速度に関して逐次代入計算を行います.
#気泡径と終末速度の初期値です.
dvs = 0.
ut = 0.
dvsnew = 0.003
epsG = 0.
Eo = 0.
Re = 0.

#Eo はエトベス数，Re は気泡レイノルズ数，CD は抗力係数
while abs((dvs - dvsnew) / dvsnew) > 0.00001:
    dvs = dvsnew
    #定義
    Eo = gra * (rhoL - rhoG) * dvs / sig

    utnew = 0.15
    while abs(((ut - utnew) / utnew)) > 0.00001:
        ut = utnew
        #定義
        Re = dvs * ut * rhoL / muL
        #抗力係数の相関
        CD = max((24 / Re * (1 + 0.15 * Re ** 0.687)), (8 / 3 * Eo / (Eo + 4)))
        #気泡の運動方程式
        utnew = (4 / 3 / CD * (1 - rhoG / rhoL) * gra * dvs) ** 0.5


    ut = utnew
    #ガスホールドアップの相関式 (二次方程式)
    Const1 = -(uG / ut) ** 0.5
    Const2 = -0.000216 * (Pv ** 0.4 * rhoL ** 0.2 / sig ** 0.6)
              * ((uG / ut) ** 0.5) * (PT / PG)
    epsG = (0.5 * (-Const1 + mt.sqrt(Const1 ** 2 - 4 * Const2))) ** 2
    #気泡径の相関
    dvsnew = 4.15 * (sig ** 0.6 / (Pv ** 0.4 * rhoL ** 0.2))
```

```
177                 * (PG / PT) * epsG ** 0.5 + 0.0009
178 dvs = dvsnew
179
180
181 #反応器内物質高さ,体積 (液相 +気相)
182 htot = hL / (1 - epsG)
183 Vtot = htot * Area
184 #体積当たり比表面積a の相関
185 a = 1.44 * ((Pv ** 0.4 * rhoL ** 0.2) / sig ** 0.6)
186     * (PT / PG) * (uG / ut) ** 0.5
187
188 #液相側物質移動係数kL の相関
189 #気泡径 0.6mm 以下の時
190 if dvs <= 0.0006:
191     kL = 0.31 * ((muL / (rhoL * DL)) ** (-2 / 3)) * ((gra * (rhoL - rhoG)
192         * muL / (rhoL ** 2)) ** (1 / 3))
193
194 #気泡系 0.6mm 以上 2.5mm 以下の時
195 elif 0.0006 < dvs <= 0.0025:
196     x = (dvs - 0.0006) / (0.0025 - 0.0006)
197     kL = (1 - x) * 0.31 * ((muL / (rhoL * DL)) ** (-2 / 3))
198         * ((gra * (rhoL - rhoG) * muL / (rhoL ** 2)) ** (1 / 3)) + x
199         * (0.42 * ((muL / (rhoL * DL)) ** (-0.5)) * ((gra * (rhoL - rhoG)
200         * muL / (rhoL ** 2)) ** (1 / 3)))
201
202 #気泡径 2.5mm 以上の時
203 else:
204     kL = 0.42 * ((muL / (rhoL * DL)) ** (-0.5)) *
205         ((gra * (rhoL - rhoG) * muL / (rhoL ** 2)) ** (1 / 3))
206
207 #液相物質移動容量係数kLa
208 kLa = kL * a
209
210 #-------------------------------------------------------------------------------
211 # 3 液相の油分について,蒸発の影響を無視小,CSTR モデルと仮定して反応器内濃度を求めます
212
213 #tol
214 C1[0] = C0[0] / (1 + (k[0] + k[1]) * tau)
215 #Bol
216 C1[1] = (C0[1] + k[0] * tau * C1[0]) / (1 + k[2] * tau)
217 #Bal
218 C1[2] = (C0[2] + (k[1] * C1[0] + k[2] * C1[1]) * tau) / (1 + k[3] * tau)
219 #Bac
220 C1[3] = C0[3] + k[3] * tau * C1[2]
221
222 #各反応の反応量
223 Rr = np.array([k[0] * C1[0], k[1] * C1[0], k[2] * C1[1], k[3] * C1[2]])
224 Rr = Rr * VL
```

```
225 #各成分の反応量
226 RR = np.array([-(Rr[0] + Rr[1]), Rr[0] - Rr[2], Rr[1] + Rr[2] - Rr[3], \
227       Rr[3], Rr[1] + Rr[2], 0., \
228       -(0.5 * Rr[0] + Rr[1] + 0.5 * Rr[2] + 0.5 * Rr[3])])
229 print(C0)
230 print(C1)
231 print(RR)
232
233
234 #%%
235 #-------------------------------------------------------------------------------
236 # 4 気相について,RK4 法を用いて蒸発量を推算します
237 # 4-first 計算に用いる関数を定義
238
239 def Gaseq(Fgas):
240 # 気相側の物質収支式
241
242     def Csatcalc(yg):
243         # 平衡濃度Csat を求める関数
244         Cs = np.array([0.] * 7)
245         xe = np.array([0.] * 7)
246
247         #tol,H2O=ラウール則,N2,O2=ヘンリー則で計算,残りは蒸発しないとする
248         xe[0] = Prea * yg[0] / 10 ** (Aant[0] - Bant[0] / (Trea + Cant[0]))
249         xe[1] = x1[1]
250         xe[2] = x1[2]
251         xe[3] = x1[3]
252         xe[4] = Prea * yg[4] / mt.exp(Aant[4] - Bant[4] / (Trea + Cant[4]))
253         xe[5] = Prea * yg[5] / HenryN2
254         xe[6] = Prea * yg[6] / HenryO2
255
256         Cs = sum(C1) * xe
257         return (Cs)
258
259     # 定義
260     Csat = np.array([0.] * 7)
261     dFgdz = np.array([0.] * 7)
262     ygas = np.array([0.] * 7)
263
264     #気相モル分率y を求め，平衡濃度 Csat を算出
265     ygas = Molefraction(Fgas)
266     Csat = Csatcalc(ygas)
267
268     #収支式の計算
269     const = kLa * pi * Drea ** 2
270     dFgdz = np.array(-const * (Csat - C1))
271
272     return (dFgdz)
```

```
273
274
275 def Molefraction(Flow):
276     fraction = np.array(Flow / sum(Flow))
277     return (fraction)
278
279
280 # 4-1 計算に用いる気液平衡データを入力
281
282 #アントワン定数 10** or exp(A-B/(T+C))
283 #tol=10** #H2O=exp
284 Aant = np.array([4.54436 + 5.0, 0., 0., 0., 23.1964, 0., 0.])
285 Bant = np.array([1738.123, 0., 0., 0., 3816.44, 0., 0.])
286 Cant = np.array([0.394, 0., 0., 0., -46.13, 0., 0.])
287
288 #ヘンリー定数 pall*y=Henry*x
289 HenryN2 = 187000000.
290 HenryO2 = 114000000.
291
292
293 # 4-2 rk4 法を用いて気相の物質収支式を解き，全体の物質収支と照らし合わせて収束させる
294
295 #窒素濃度の初期値.これは蒸発も溶解もしないように収束させる
296 C1[5] = 0.5
297 #水と酸素の初期値.これらは反応生成量,消費量が蒸発 (溶解)量と等しくなるように収束させる
298 C1[4] = 10.
299 C1[6] = 0.5
300
301 #誤差評価関数初期値
302 eps = 1000.
303
304 #ステップ数
305 step = 50
306 #刻み幅
307 h = htot / step
308
309 j = 0
310 while eps >= 0.001 and j <= 100:
311     j = j + 1
312     x1 = Molefraction(C1)
313     #計算用
314     Fg = F2.copy()
315
316     for i in range(0, step - 1):
317
318         kk1 = np.array(h * Gaseq(Fg))
319         kk2 = np.array(h * Gaseq(Fg + kk1 / 2.))
320         kk3 = np.array(h * Gaseq(Fg + kk2 / 2.))
```

```
        kk4 = np.array(h * Gaseq(Fg + kk3))

        Fg += (kk1 + 2. * (kk2 + kk3) + kk4) / 6.
    next

    #結果
    F3 = Fg.copy()

    #誤差を求める
    epslist = np.array([((F3[4] - RR[4]) / RR[4]) ** 2, \
            ((F2[5] - F3[5]) / F2[5]) ** 2, \
            (((F3[6] - F2[6]) - RR[6]) / RR[6]) ** 2])
    eps = sum(epslist)

    print(eps)

    #値の改善

    C1[4] += - (F3[4] - RR[4])
    C1[5] += (F2[5] - F3[5])
    C1[6] += - ((F3[6] - F2[6]) - RR[6])


# 4-3 収束時の分布を取得

#-------------------------------------------------------------------------------
# 5 デカンター周りの各流量を求め,液相の反応に対して蒸発の影響が無視できるか確認する
# 5-1 各流量を算出する.
#F5 について，デカンターで Tdec まで蒸気 F3 が冷やされ，水とトルエンが飽和状態まで凝縮する.
y3 = Molefraction(F3)
F5 = F3.copy()
y5sat = np.array([10 ** (Aant[0] - Bant[0] / (Tdec + Cant[0])) / Prea, 0, \
        0, 0, mt.exp(Aant[4] - Bant[4] / (Tdec + Cant[4])) / Prea, 0, 0])
F5[0] = y5sat[0] / (1 - y5sat[0] - y5sat[4]) * (F5[5] + F5[6])
F5[4] = y5sat[4] / (1 - y5sat[0] - y5sat[4]) * (F5[5] + F5[6])
y5 = Molefraction(F5)


#溶解度が低いため,デカンターにおける分離効率は近似的に 100%とする
F4[0] = F3[0] - F5[0]

#冷却のため投入する水量F6 を計算する


#F7 を計算
F7[4] = F6[4] + F3[4] - F5[4]


```

```
369 # 6 結果を出力する
370
371 wb = px.load_workbook(conclusionbookname)
372 ws = wb.active
373
374 for i in range(7):
375     ws.cell(row=i + 2, column=2).value = F2[i]
376     ws.cell(row=i + 2, column=3).value = F3[i]
377     ws.cell(row=i + 2, column=4).value = F4[i]
378     ws.cell(row=i + 2, column=5).value = F5[i]
379
380     ws.cell(row=i + 10, column=1).value = x1[i]
381     ws.cell(row=i + 10, column=2).value = y2[i]
382     ws.cell(row=i + 10, column=3).value = y3[i]
383     ws.cell(row=i + 10, column=4).value = y4[i]
384     ws.cell(row=i + 10, column=5).value = y5[i]
385
386 wb.save(filename=conclusionbookname)
```

迅速に気相が平衡に達すると仮定した場合の上記プログラムは以下のようになり，実際の解析ではこちらを用いた.

ソースコード B.2　気液反応器の解析 (迅速に平衡)

```
1   # 気液反応器の解析をします.特に断りがなければSI 単位です
2
3   # FlowNO : 0=反応器への流入液，1=反応器からの流出液，2=反応器への流入空気，
4   # 3=反応からデカンターへの蒸気
5   # 4=反応器への還流油分，5=デカンターからの流出空気,
6   # 6=デカンターに入れる冷却水，7=デカンターからの流出水
7   # ComponentNO: 0=Tol，1=Bol，2=Bal，3=Bac，4=H2O，5=N2，6=O2
8   # ver_slim : 迅速な平衡の仮定から各流量を求めるslim な program です
9   import math
10  import numpy as np
11  import xlwings as xw
12
13  def main():
14
15      # #-----------------Excel シートからdata を取得-----------------------------
16      wb = xw.Book.caller()
17      F0_input = wb.sheets[0].range((43, 4), (43, 10)).value
18                                                  # 反応器入口成分別モル流量
19      P_input = wb.sheets[0].range('D3').value # 反応器圧力
20      VL_input = wb.sheets[0].range('D4').value # 反応器体積
21      T_input = wb.sheets[0].range('D5').value # 反応器温度
22      T_decanter = wb.sheets[0].range('D7').value # デカンタ温度
23      v0_input = wb.sheets[0].range('N43').value # 反応器入口体積流量
24      topRow = 46
25      # ------------------------------------------------------------------------
26
```

```
27      # 流れのクラスと流れに関係するメソッド
28      class Flow:
29
30          # 中身
31          def __init__(self):
32              # モル流量,モル分率,濃度,平衡での気液モル分率,温度を持っています
33              self.MolerFlow = np.array([0.]*7)
34              self.Fraction = np.array([0.]*7)
35              self.Concentration = np.array([0.]*7)
36              self.EquilibriumGasFraction = np.array([0.]*7)
37              self.EquilibriumLiquidFraction = np.array([0.]*7)
38              self.MolerEnthalpy = np.array([0.]*7)
39              self.Enthalpy = np.array([0.]*7)
40              self.EnthalpyTotal = 0.0
41              self.Temp = 0.0
42
43          # アントワン定数 10** or exp(A-B/(T+C))
44          # tol=10** H2O=exp
45          antoineA = np.array([4.54436 + 5.0, 0., 0., 0., 23.1964, 0., 0.])
46          antoineB = np.array([1738.123, 0., 0., 0., 3816.44, 0., 0.])
47          antoineC = np.array([0.394, 0., 0., 0., -46.13, 0., 0.])
48
49          # ヘンリー定数 pall*y=Henry*x
50          HenryN2 = 187000000.
51          HenryO2 = 114000000.
52
53          # 標準モル生成エンタルピー
54          LiquidDeltafEnthalpy = np.array([12.4 ,-94.1 ,-87.0 ,-385.2 \
55                                      ,-285.83 ,0. ,0. ])
56          GasDeltafEnthalpy = np.array([50.4 ,154.9 ,-36.8 ,-290.4 \
57                                      ,-241.818 ,0. ,0. ])
58
59          # 熱容量係数
60          GasheatcapacityA = np.array([-24.356 , 0. , 0. ,-51.295 \
61                                      ,32.244 ,28.107 ,24.234 ])
62          GasheatcapacityB = np.array([0.513 , 0. , 0. , 0.629 \
63                                      ,1.92e-3 ,-3.68e-6 ,4.84e-3 ])
64          GasheatcapacityC = np.array([-2.77e-5 , 0. , 0. , -4.24e-5 \
65                                      ,1.06e-5 ,1.75e-5 ,-2.08e-5 ])
66          GasheatcapacityD = np.array([4.91e-8 , 0. , 0. , 1.06e-7 \
67                                      ,-3.60e-9 ,-1.07e-8 ,2.93e-10])
68          # 熱容量係数
69          LiqheatcapacityA = np.array([157.09 , 0. , 0. ,0. \
70                                      ,75.375 , 0. , 0. ])
71          LiqheatcapacityB = np.array([0. , 0. , 0. ,0. \
72                                      ,0. , 0. , 0. ])
73          LiqheatcapacityC = np.array([0. , 0. , 0. ,0. \
74                                      ,0. , 0. , 0. ])
```

```
75          LiqheatcapacityD = np.array([0. , 0. , 0. ,0. \
76                                      ,0. , 0. , 0. ])
77
78
79          # モル流量からモル分率を求めるメソッド
80          def FractionCalc(self):
81              FlowSum = np.sum(self.MolerFlow)
82              self.Fraction = self.MolerFlow/FlowSum
83
84          # 濃度からモル分率を求めるメソッド
85
86          def FractionCalcFromConcentration(self):
87              FlowSum = np.sum(self.Concentration)
88              self.Fraction = self.Concentration/FlowSum
89
90          # decanter で Tol と水の平衡状態の気体モル分率を求めるメソッド
91
92          def DecanterEquilibriumGasFractionCalc(self):
93              self.EquilibriumGasFraction[0] = (
94                  10 ** (self.antoineA[0] - self.antoineB[0] /
95                  (Tdec + self.antoineC[0])))/Prea
96              self.EquilibriumGasFraction[4] = (
97                  math.exp(self.antoineA[4] - self.antoineB[4] /
98                  (Tdec + self.antoineC[4])))/Prea
99
100         # CSTR(液相)の設計方程式から油分の液相濃度を求めさせるメソッド
101         def CSTRLiquidCalc(self):
102             # tol
103             Flow1.Concentration[0] = Flow0.Concentration[0] /
104                                      (1 + (k[0] + k[1]) * tau)
105             # Balc
106             Flow1.Concentration[1] = (Flow0.Concentration[1] + \
107                       k[0] * tau * Flow1.Concentration[0]) / (1 + k[2] * tau)
108             # Bald
109             Flow1.Concentration[2] = (Flow0.Concentration[2] + \
110                   (k[1] * Flow1.Concentration[0] + \
111                    k[2] * Flow1.Concentration[1]) * tau) / (1 + k[3] * tau)
112             # BzA
113             Flow1.Concentration[3] = Flow0.Concentration[3] + \
114                                      k[3] * tau * Flow1.Concentration[2]
115
116         # reactor で平衡状態の液モル分率を求めるメソッド
117         def EquilibriumLiquidFractionCalc(self):
118             self.EquilibriumLiquidFraction[0] = Prea * self.Fraction[0] /
119                                  10 ** (self.antoineA[0] - self.antoineB[0] /
120                                  (Trea + self.antoineC[0]))
121             self.EquilibriumLiquidFraction[1] = Flow1.Fraction[1]
122             self.EquilibriumLiquidFraction[2] = Flow1.Fraction[2]
```

```
123             self.EquilibriumLiquidFraction[3] = Flow1.Fraction[3]
124             self.EquilibriumLiquidFraction[4] = Prea * self.Fraction[4] /
125                                 math.exp(self.antoineA[4] - self.antoineB[4] /
126                                (Trea + self.antoineC[4]))
127             self.EquilibriumLiquidFraction[5] = Prea * self.Fraction[5] /
128                                                 self.HenryN2
129             self.EquilibriumLiquidFraction[6] = Prea * self.Fraction[6] /
130                                                 self.HenryO2
131
132         # 気流が迅速に平衡に達すると仮定してほかの流量や濃度を求めるメソッド
133         def EquilibriumAssumptionCalc(self):
134
135             Flow1.FractionCalcFromConcentration()
136
137             # F2 から F3 の決定
138             Flow3.MolerFlow[4] = RR[4]
139             Flow3.MolerFlow[5] = Flow2.MolerFlow[5]
140             Flow3.MolerFlow[6] = Flow2.MolerFlow[6] + RR[6]
141
142             Flow3.Fraction[0] = Flow1.Fraction[0] /
143                                 Prea * 10. ** \
144                                 (self.antoineA[0] - self.antoineB[0] / \
145                                 (Trea + self.antoineC[0]))
146             sumF3 = np.sum(Flow3.MolerFlow)
147             Flow3.MolerFlow[0] = Flow3.Fraction[0] /
148                                 (1-Flow3.Fraction[0])*(sumF3)
149             Flow3.FractionCalc()
150
151             #迅速に平衡に達すると仮定してF1 の H2O,N2,O2 濃度を決定
152             Flow3.EquilibriumLiquidFractionCalc()
153             C1sum=np.sum(Flow1.Concentration)
154             Flow1.Fraction[4] = Flow3.EquilibriumLiquidFraction[4]
155             Flow1.Fraction[5] = Flow3.EquilibriumLiquidFraction[5]
156             Flow1.Fraction[6] = Flow3.EquilibriumLiquidFraction[6]
157             Flow1.Concentration[4] = Flow1.Fraction[4] /
158                                      (1-Flow1.Fraction[4]) * C1sum
159             Flow1.Concentration[5] = Flow1.Fraction[5] /
160                                      (1-Flow1.Fraction[5]) * C1sum
161             Flow1.Concentration[6] = Flow1.Fraction[6] /
162                                      (1-Flow1.Fraction[6]) * C1sum
163             Flow1.FractionCalcFromConcentration()
164             Flow1.MolerFlow = Flow1.Concentration * v0
165
166             # F5 について，デカンターで Tdec まで蒸気 F3 が冷やされ，
167             # 水とトルエンが飽和状態まで凝縮するとして決定
168             Flow5.MolerFlow = np.copy(Flow3.MolerFlow)
169             Flow5.DecanterEquilibriumGasFractionCalc()
170             Flow5.MolerFlow[0] = Flow5.EquilibriumGasFraction[0] /
```

```
                        (1 - Flow5.EquilibriumGasFraction[0] - \
                        Flow5.EquilibriumGasFraction[4]) * \
                        (Flow5.MolerFlow[5] + Flow5.MolerFlow[6])
    Flow5.MolerFlow[4] = Flow5.EquilibriumGasFraction[4] /
                        (1 - Flow5.EquilibriumGasFraction[0] - \
                        Flow5.EquilibriumGasFraction[4]) * \
                        (Flow5.MolerFlow[5] + Flow5.MolerFlow[6])
    Flow5.FractionCalc()

    # 溶解度が低いため,デカンターにおける分離効率は近似的に 100%としてF4 を決定
    Flow4.MolerFlow[0] = Flow3.MolerFlow[0] - Flow5.MolerFlow[0]
    #F1 に蒸発の寄与を取り込む
    Flow1.MolerFlow[0] += -Flow5.MolerFlow[0]

    # 冷却のため投入する水量F6 を決定
    #モルエンタルピーの計算
    Flow1.LiquidEnthalpyCalc()
    Flow2.GasEnthalpyCalc()
    Flow3.GasEnthalpyCalc()
    Flow4.LiquidEnthalpyCalc()
    Flow5.GasEnthalpyCalc()
    Flow6.LiquidEnthalpyCalc()
    Flow7.LiquidEnthalpyCalc()

    #連立方程式を解きF6 と F7 を求める
    A1list=[1,-1]
    A2list=[Flow6.MolerEnthalpy[4] , -Flow7.MolerEnthalpy[4]]
    A_matrix = np.array([A1list, A2list])
    b = np.array([Flow4.MolerFlow[4] + Flow5.MolerFlow[4] - \
        Flow3.MolerFlow[4] ,Flow4.EnthalpyTotal + \
        Flow5.EnthalpyTotal - Flow3.EnthalpyTotal ])
    x_vec = np.linalg.solve(A_matrix, b)
    Flow6.MolerFlow[4] = x_vec[0]
    Flow7.MolerFlow[4] = x_vec[1]


# 気体のモルエンタルピーを求めるメソッド [J/mol]
def GasEnthalpyCalc(self):
    self.MolerEnthalpy = self.GasDeltafEnthalpy*1000+
    self.GasheatcapacityA*(self.Temp -298.15 ) + \
    self.GasheatcapacityB*(self.Temp**2.-298.15**2.)/2. + \
    self.GasheatcapacityC*(self.Temp**3.-298.15**3.)/3. + \
    self.GasheatcapacityD*(self.Temp**4.-298.15**4.)/4.
    self.Enthalpy = self.MolerFlow * self.MolerEnthalpy
    self.EnthalpyTotal=np.sum(self.Enthalpy)

# 液体のモル当たりエンタルピーを求めるメソッド [J/mol]
def LiquidEnthalpyCalc(self):
```

```
219             self.MolerEnthalpy = self.LiquidDeltafEnthalpy*1000 + \
220                                  self.LiqheatcapacityA*(self.Temp-298.15)
221             self.Enthalpy = self.MolerFlow * self.MolerEnthalpy
222             self.EnthalpyTotal=np.sum(self.Enthalpy)
223
224     # インスタンスの定義
225     Flow0 = Flow()
226     Flow1 = Flow()
227     Flow2 = Flow()
228     Flow3 = Flow()
229     Flow4 = Flow()
230     Flow5 = Flow()
231     Flow6 = Flow()
232     Flow7 = Flow()
233
234     # ------------------外部入力値-----------------------------------------------
235
236     # 0．外部入力条件を受けとります
237     # 液相体積
238     VL = VL_input
239     # 流入液体積 [m3/s]
240     v0 = v0_input # [m3/hour]
241     v0 = v0 / 3600.
242     # 反応器体積
243     DHrate = 1.0
244     # 全圧 [Pa]
245     Prea = P_input * 1.e+5
246     # 温度 [℃]
247     Trea = T_input + 273.15
248     # 流入液モル流量 [mol/s]
249     Flow0.MolerFlow = np.array(F0_input) # kmol/hour
250     Flow0.MolerFlow = Flow0.MolerFlow / 3.6 # mol/s
251     # ------------------------------------------------------------------------
252
253     # 1 入力 ----------------------------------------------------------------
254     # 1-1 計算に用いる物性パラメータを入力します
255     # 自然定数
256     pi = math.pi
257     Rgas = 8.314
258     gra = 9.81
259
260     # アレニウス型反応速度式のパラメータ
261     # 頻度因子 [s-1]
262     A = np.array([math.exp(20.634), math.exp(17.928),
263                   math.exp(15.4), math.exp(19.698)])
264     A = A / 3600.
265     # 活性化エネルギー
266     E = np.array([81.389, 69.53, 56.987, 71.442])
```

```
267         E = E * 1000.
268         # 反応速度定数 [s-1]
269         k = np.array([A[i] * math.exp(-E[i] / (Rgas * Trea)) for i in range(4)])
270
271         # 1-2 反応器の装置条件を決めます
272         Drea = (VL / (pi * DHrate)) ** (1 / 3)
273         hL = Drea * DHrate
274         Area = pi * (Drea ** 2)/4.
275         Dnoz = Drea / 10
276         Anoz = pi * (Dnoz ** 2)/4.
277
278         # 1-3 反応器の運転条件を決めます
279         tau = VL / v0
280         QG = (50. / 1000. / 3600.) * (VL / 0.00025)*0.6
281         Tdec = T_decanter + 273.15
282
283
284         #global 宣言
285         kLa = 0.0
286         htot = 0.0
287         Rr = np.array([0.]*4)
288         RR = np.array([0.]*7)
289
290
291         # 1-4 マテリアルフローに必要な変数を入力します
292
293         Twater = 25.0
294
295         Flow2.Temp = 10. + 273.15
296         Flow3.Temp = Trea
297         Flow4.Temp = Tdec
298         Flow5.Temp = Tdec
299         Flow6.Temp = Twater + 273.15
300         Flow7.Temp = Tdec
301
302         # 各フローのモル分率
303         Flow0.FractionCalc()
304         Flow2.Fraction = np.array([0, 0, 0, 0, 0, 0.79, 0.21])
305         Flow4.Fraction = np.array([1., 0, 0, 0, 0, 0, 0])
306         Flow6.Fraction = np.array([0., 0., 0., 0., 1., 0., 0.])
307         Flow7.Fraction = np.array([0, 0, 0, 0, 1., 0, 0])
308
309         # 各フローのモル流量
310         Flow2.MolerFlow = np.array([0., 0., 0., 0., 0., ((Prea * 0.79) / \
311               (Rgas * Flow2.Temp)), ((Prea * 0.21) / (Rgas * Flow2.Temp))])
312         Flow2.MolerFlow = Flow2.MolerFlow * QG
313
314         # 各フローのモル濃度
```

```
315     Flow0.Concentration = Flow0.MolerFlow / v0
316
317     # 2 液相の油分について,蒸発の影響を無視小,
318     # CSTR モデルと仮定して反応器内濃度を求めます ------------------------
319
320     Flow1.CSTRLiquidCalc()
321
322     # 各反応の反応量
323     Rr = np.array([k[0] * Flow1.Concentration[0], \
324                    k[1] * Flow1.Concentration[0], \
325                    k[2] * Flow1.Concentration[1], \
326                    k[3] * Flow1.Concentration[2]])
327     Rr = np.array(Rr * VL)
328     # 各成分の反応量
329     RR = np.array([-(Rr[0] + Rr[1]), \
330                Rr[0] - Rr[2], \
331                Rr[1] + Rr[2] - Rr[3], \
332                Rr[3],Rr[1] + Rr[2], \
333                0.,\
334                -(0.5 * Rr[0] + Rr[1] + 0.5 * Rr[2] + 0.5 * Rr[3])])
335
336
337     # 3 気相について,平衡に達すると仮定してF3 や他の流量ををを推算します -----------
338     Flow3.EquilibriumAssumptionCalc()
339
340
341     #--------------------------------出力-------------------------------------
342
343     H7_6_cooling = Flow6.MolerFlow[4] *
344                  (Flow7.MolerEnthalpy[4] - Flow6.MolerEnthalpy[4])
345                  /1000. /1000. *3600
346     H6_1000kmol_cooling = 1000.* 1000. / 3600.*
347                  (Flow7.MolerEnthalpy[4] - Flow6.MolerEnthalpy[4])
348                  /1000. /1000. *3600
349
350     if H7_6_cooling > H6_1000kmol_cooling:
351         Flow6.MolerFlow[4] = 1000. * 1000. / 3600.
352         Flow7.MolerFlow[4] = Flow6.MolerFlow[4] -
353                              Flow5.MolerFlow[4] +
354                              Flow3.MolerFlow[4]
355
356     Flow6.LiquidEnthalpyCalc()
357     Flow7.LiquidEnthalpyCalc()
358
359     Flow6.Temp += -273.15
360     Flow7.Temp += -273.15
361     F1_output = Flow1.MolerFlow * 3.6 # 反応器出口成分別モル流量
362     F2_output = Flow2.MolerFlow * 3.6 # 反応器出口成分別モル流量
```

```
363     F3_output = Flow3.MolerFlow * 3.6 # 反応器出口成分別モル流量
364     F4_output = Flow4.MolerFlow * 3.6 # 反応器出口成分別モル流量
365     F5_output = Flow5.MolerFlow * 3.6 # 反応器出口成分別モル流量
366     F6_output = Flow6.MolerFlow * 3.6 # 反応器出口成分別モル流量
367     F7_output = Flow7.MolerFlow * 3.6 # 反応器出口成分別モル流量
368     Flow1.EnthalpyTotal = Flow1.EnthalpyTotal * 1.e-9
369     Flow2.EnthalpyTotal = Flow2.EnthalpyTotal * 1.e-9
370     Flow3.EnthalpyTotal = Flow3.EnthalpyTotal * 1.e-9
371     Flow4.EnthalpyTotal = Flow4.EnthalpyTotal * 1.e-9
372     Flow5.EnthalpyTotal = Flow5.EnthalpyTotal * 1.e-9
373     Flow6.EnthalpyTotal = Flow6.EnthalpyTotal * 1.e-9
374     Flow7.EnthalpyTotal = Flow7.EnthalpyTotal * 1.e-9
375
376     wb.sheets[0].range((44, 4), (44, 10)).value = F1_output # 流出液流量
377     wb.sheets[0].range(44, 12).value = Flow1.EnthalpyTotal # 流出液流量
378
379     wb.sheets[0].range((topRow + 0, 4), (topRow + 0, 10)).value = F2_output
380                                                 # 流入空気流量
381     wb.sheets[0].range((topRow + 1, 4), (topRow + 1, 10)).value = F3_output
382                                                 # 反応器からデカンターへの蒸気
383     wb.sheets[0].range((topRow + 2, 4), (topRow + 2, 10)).value = F4_output
384                                                 # 反応器へ還流される油分
385     wb.sheets[0].range((topRow + 3, 4), (topRow + 3, 10)).value = F5_output
386                                                 # 吸着装置へ運ばれるTol 飽和蒸気
387     wb.sheets[0].range((topRow + 4, 4), (topRow + 4, 10)).value = F6_output
388                                                 # 冷却のため投入する水量
389     wb.sheets[0].range((topRow + 5, 4), (topRow + 5, 10)).value = F7_output
390                                                 # デカンターからパージする水量
391     wb.sheets[0].range((topRow + 6, 4), (topRow + 6, 10)).value
392                                                 = Flow1.Concentration
393                                                 # 流出液流量
394
395     wb.sheets[0].range(topRow + 0, 15).value = Flow2.EnthalpyTotal
396                                                 # 流入空気流量
397     wb.sheets[0].range(topRow + 1, 15).value = Flow3.EnthalpyTotal
398                                                 # 反応器からデカンターへの蒸気
399     wb.sheets[0].range(topRow + 2, 15).value = Flow4.EnthalpyTotal
400                                                 # 反応器へ還流される油分
401     wb.sheets[0].range(topRow + 3, 15).value = Flow5.EnthalpyTotal
402                                                 # 吸着装置へ運ばれるTol 飽和蒸気
403     wb.sheets[0].range(topRow + 4, 15).value = Flow6.EnthalpyTotal
404                                                 # 冷却のため投入する水量
405     wb.sheets[0].range(topRow + 5, 15).value = Flow7.EnthalpyTotal
406                                                 # デカンターからパージする水量
407
408     wb.sheets[0].range(74,6).value = H7_6_cooling
409     wb.sheets[0].range(74,7).value = Flow6.Temp
410     wb.sheets[0].range(74,8).value = Flow7.Temp +0.001
```

```
411
412         wb.sheets[0].range(75,6).value = H6_1000kmol_cooling
413         wb.sheets[0].range(75,7).value = Twater
414         wb.sheets[0].range(75,8).value = T_decanter
415
416         wb.sheets[0].range(76,6).value = H6_1000kmol_cooling
417         wb.sheets[0].range(76,7).value = T_decanter
418         wb.sheets[0].range(76,8).value = Twater
```

ソースコード B.3　グランドコンポジットカーブおよび TQ 線図を書くコード

```
1  # ================================================================================
2  # プログラム: 熱交換の解析
3  #
4  # ver010
5  # : グランドコンポジットカーブを描画します.
6  # ver020
7  # : 追加 TQ 線図を描画します
8  # ver030
9  # : 追加 各流体の熱交換対応を解析します
10 # ver040
11 # : 追加 任意の等温外部熱媒の熱量計算を実装
12 #
13 # 使い方:
14 # xlwings をimport したエクセルファイルに,
15 # 相状態 [気体,液体,凝縮ガス,蒸気液] = ["gas","liq","congas","vapliq"]
16 # を書き込む
17 #（一部データを入れていない総括伝熱係数の組み合わせがある）
18 # 等温流体には"give","receive"を明示する.
19 #（エラーを吐くときは人為的に微小な温度勾配をつける）
20 # 外部熱媒には"exgive","exreceive"と書く
21 # 横の列に交換熱量,熱交換前温度,熱交換後の温度の順に書き込む.
22 # 熱交換流体の総数を書き込む.（外部熱媒を含む）
23 # 最小接近温度差を決める.
24 # vba から Runpython コマンドで実行 .
25 #
26 # ================================================================================
27
28 def main():
29
30 #ライブラリ
31 from scipy import interpolate
32 from math import log
33 import numpy as np
34 import xlwings as xw
35 import copy
36 import sys
37 from matplotlib import pyplot as plt
38 from matplotlib.font_manager import FontProperties
```

```
39 from matplotlib import rcParams
40
41
42 wb = xw.Book.caller()
43
44 #==================================外部入力値====================================
45 readsheet = 0 #読み込みたいシートNO(0始まり)
46 writesheet = 1 #書きこみたいシートNO(0始まり)
47 row_readstart = 64 #読み込みたい表の左上の行
48 column_readstart = 3 #読み込みたい表の左上の列
49 row_writestart = 2 #書き込みたい表の左上の行
50 column_writestart = 2 #書き込みたい表の左上の列
51 #================================================================================
52
53 sn_input = wb.sheets[readsheet].range((row_readstart-1, column_readstart-1 ))
54               .value
55 mdt_input = wb.sheets[readsheet].range((row_readstart, column_readstart-1 ))
56               .value
57
58 streamnumber = int(sn_input) #熱交換流体の総数
59 minimumdeltaTemperture = float(mdt_input) #最小接近温度差
60
61 sig = 1 #切り捨てる少数点以下桁数
62
63 #識別用
64 #授受関係
65 give = "give" #与熱流体
66 receive = "receive" #受熱流体
67 externalgive = "exgive" #外部熱媒
68 externalreceive = "exreceive" #外部冷媒
69 gives = [give ,externalgive ]
70 receives = [receive,externalreceive]
71
72 #相状態関係
73 gas = "gas"
74 liquid = "liq"
75 condensategas = "congas"
76 vaporliquid = "vapliq"
77 phaselist = [gas, liquid, condensategas, vaporliquid]
78
79 twophaselist = []
80 for i in range (len(phaselist)):
81     for j in range (len(phaselist)):
82         if i <= j:
83             twophaselist.append([phaselist[i], phaselist[j]])
84     next
85 next
86
```

```
87 #overall_heattransfer_coefficient = 総括伝熱係数U[W/m2 K]
88 # [gasgas],[gas liq],[gas congas],[gas vapgas],[liqliq],
89 # [liq congas],[liq vapliq],[congascongas],[congas vapliq][vapliqvapliq]
90 Ulist = [150., 200., 500., 500., 300., 1000., 1000., "none", 1500.,"none" ]
91
92 #float 判定用
93 eps = 10.**( - float(sig))
94
95 #熱媒検索用
96 streamindex_give =[]
97 streamindex_receive =[]
98 streamindex_externalgive =[]
99 streamindex_externalreceive =[]
100 streamindex_externalheat =[]
101
102 class Heatstream():
103
104     def __init__(self, number):
105         self.name ="" #ストリーム名
106         self.property =""
107                                     #(give or receive) or (exgive or exreceive)
108         self.phase ="" #状態
109         self.number = number #ストリーム番号
110         self.deltaheat = 0.0 #交換熱量
111         self.deltaheat_section = 0.0 #区間交換熱量
112         self.heatcapacity = 0.0 #熱容量
113         self.temperatureMAX = 0.0 #高温
114         self.temperatureMIN = 0.0 #低温
115         self.temperatureindexMAX = 0 #高温検索用
116         self.temperatureindexMIN = 0 #低温検索用
117         self.rebcon ="" #reboier 判別用
118
119         self.leftend_externalheat_GCC =0.0 #外部熱媒用
120
121     def read(self):
122         # excel 連携用
123         self.name = wb.sheets[readsheet]
124           .range((row_readstart + self.number + 1 ,column_readstart + 0 )).value
125         self.phase = wb.sheets[readsheet]
126           .range((row_readstart + self.number + 1 ,column_readstart + 1 )).value
127         self.property = wb.sheets[readsheet]
128           .range((row_readstart + self.number + 1 ,column_readstart + 2 )).value
129         self.deltaheat = wb.sheets[readsheet]
130           .range((row_readstart + self.number + 1 ,column_readstart + 3 )).value
131         self.temperatureMAX = wb.sheets[readsheet]
132           .range((row_readstart + self.number + 1 ,column_readstart + 4 )).value
133         self.temperatureMIN = wb.sheets[readsheet]
134           .range((row_readstart + self.number + 1 ,column_readstart + 5 )).value
```

```
135         self.rebcon = wb.sheets[readsheet]
136           .range((row_readstart + self.number + 1 ,column_readstart + 6 )).value
137
138
139         #ねんのため
140         self.name = str(self.name)
141         self.phase = str(self.phase)
142         self.property = str(self.property)
143         self.deltaheat = float(self.deltaheat)
144         self.temperatureMAX = float(self.temperatureMAX)
145         self.temperatureMIN = float(self.temperatureMIN)
146
147         ddmax = np.random.rand() * 0.0001 + 0.0001 #等温熱媒調整用微小項
148         ddmin = np.random.rand() * 0.0001 + 0.0001 #等温熱媒調整用微小項
149
150         if (self.temperatureMAX - self.temperatureMIN) > 0 :#与熱流体
151             self.property = give
152             streamindex_give.append(self.number)
153
154         elif (self.temperatureMAX - self.temperatureMIN) < 0 :#受熱流体
155             self.property = receive
156             self.temperatureMAX, self.temperatureMIN
157                                 = self.temperatureMIN, self.temperatureMAX
158             streamindex_receive.append(self.number)
159
160         self.temperatureMAX += ddmax #傾きがないとエラーなので微小勾配をつける
161         self.temperatureMIN += -ddmin
162
163         if self.property == externalgive :#外部与熱流体
164             self.deltaheat = 0.0
165             streamindex_externalgive.append(self.number)
166
167             streamindex_externalheat.append(self.number)
168
169         elif self.property == externalreceive :#外部受熱流体
170             self.deltaheat = 0.0
171             streamindex_externalreceive.append(self.number)
172
173             streamindex_externalheat.append(self.number)
174
175         self.heatcapacity_calc()
176
177     def heatcapacity_calc(self):
178         self.heatcapacity = self.deltaheat
179                         / (self.temperatureMAX - self.temperatureMIN)
180
181
182 streams=[] #熱流体の情報読み込み
```

```
for i in range (streamnumber):
    dummy = Heatstream(i)
    streams.append(dummy)
    streams[i].read()

ascendingtemperature_TQ=[] #温度の昇順ソート
ascendingtemperature_GCC=[]
for i in range (len(streams)):
    ascendingtemperature_TQ.append(streams[i].temperatureMAX) #TQ 線図用
    ascendingtemperature_TQ.append(streams[i].temperatureMIN)
    if streams[i].property == give: #GCC 用
        ascendingtemperature_GCC.append(streams[i].temperatureMAX)
        ascendingtemperature_GCC.append(streams[i].temperatureMIN)
    elif streams[i].property == receive:
        ascendingtemperature_GCC
            .append(streams[i].temperatureMAX + minimumdeltaTemperture)
        ascendingtemperature_GCC
            .append(streams[i].temperatureMIN + minimumdeltaTemperture)
next

ascendingtemperature_TQ.sort()
ascendingtemperature_GCC.sort()

#整理した温度のindex を取得
for i in range (len(streams)):
    streams[i].temperatureindexMAX = ascendingtemperature_TQ
                        .index(streams[i].temperatureMAX)
    streams[i].temperatureindexMIN = ascendingtemperature_TQ
                        .index(streams[i].temperatureMIN)

heatbalance_section_GCC =np.array([0.]*(len(ascendingtemperature_GCC)))
                                                        #各温度範囲での熱バランス
heatbalance_sum_GCC =np.array([0.]*(len(ascendingtemperature_GCC)))
                                                        #GCC グラフ用の熱量

for i in range (len(ascendingtemperature_GCC) - 1): #GCC カーブデータの作成
    dT = (ascendingtemperature_GCC[i+1] - ascendingtemperature_GCC[i])
    for j in range (len(streamindex_give)):
        x = streams[streamindex_give[j]]
        if x.temperatureMIN <= ascendingtemperature_GCC[i] < x.temperatureMAX:
            heatbalance_section_GCC[i] += + x.heatcapacity * dT
    for j in range (len(streamindex_receive)):
        x = streams[streamindex_receive[j]]
        if (x.temperatureMIN + minimumdeltaTemperture) \
            <=ascendingtemperature_GCC[i] \
            < (x.temperatureMAX + minimumdeltaTemperture):
            heatbalance_section_GCC[i] += - x.heatcapacity * dT

```

```
231     heatbalance_sum_GCC[i+1] = heatbalance_sum_GCC[i] - \
232                                heatbalance_section_GCC[i]
233 next
234
235 GCCleft = np.min(heatbalance_sum_GCC) #カーブ左端の探索
236 heatbalance_sum_GCC = heatbalance_sum_GCC - GCCleft #GCC カーブをずらす
237
238 #=============================================================================
239 #外部熱媒使用量の計算
240
241 GCCfunction = interpolate.interp1d(ascendingtemperature_GCC,heatbalance_sum_GCC)
242
243 def ExternalHeatFromGCCfunction(EXheatstream): #外部熱媒の使用可能量を計算する関数
244     EXHeat = 0.0
245
246     if EXheatstream.property == externalgive: #熱媒の場合
247         if EXheatstream.temperatureMIN >= ascendingtemperature_GCC[-1]:
248             EXHeat = heatbalance_sum_GCC[-1]
249         else :
250             EXHeat = GCCfunction(EXheatstream.temperatureMIN)
251             for i in range(len(ascendingtemperature_GCC)):
252                 if ascendingtemperature_GCC[i] >= EXheatstream.temperatureMIN:
253                     EXHeat = np.min([EXHeat, heatbalance_sum_GCC[i]])
254         next
255
256     if EXheatstream.property == externalreceive: #冷媒の場合
257         if EXheatstream.temperatureMIN + minimumdeltaTemperture \
258             <= ascendingtemperature_GCC[0]:
259             EXHeat = heatbalance_sum_GCC[0]
260         else :
261             EXHeat = GCCfunction(EXheatstream.temperatureMIN + \
262                                  minimumdeltaTemperture)
263             for i in range(len(ascendingtemperature_GCC)):
264                 if ascendingtemperature_GCC[i]
265                    <=EXheatstream.temperatureMIN + minimumdeltaTemperture:
266                     EXHeat = np.min([EXHeat, heatbalance_sum_GCC[i]])
267         next
268
269     EXheatstream.deltaheat = EXHeat
270
271
272 for i in range (len(streamindex_externalheat)): #外部熱媒の使用可能量を計算
273     ExternalHeatFromGCCfunction(streams[streamindex_externalheat[i]])
274
275
276 for i in range (len(streamindex_externalgive)):
277             #温度の異なる外部熱媒を用いている時,熱量を分割する処理
278     for j in range (len(streamindex_externalgive)):
```

```
279             #低コスト熱媒に代用可能な部分の検索
280         if (streams[streamindex_externalgive[i]].temperatureMIN \
281           > streams[streamindex_externalgive[j]].temperatureMIN):
282                streams[streamindex_externalgive[i]].leftend_externalheat_GCC = \
283                     np.max([streams[streamindex_externalgive[i]]
284                         .leftend_externalheat_GCC, \
285                         streams[streamindex_externalgive[j]].deltaheat])
286
287
288 for i in range (len(streamindex_externalreceive)):
289             #温度の異なる外部冷媒を用いている時,熱量を分割する処理
290     for j in range (len(streamindex_externalreceive)):
291             #低コスト熱媒に代用可能な部分の検索
292         if streams[streamindex_externalreceive[i]].temperatureMIN
293             < streams[streamindex_externalreceive[j]].temperatureMIN:
294                 streams[streamindex_externalreceive[i]] \
295                       .leftend_externalheat_GCC = \
296                      np.max([streams[streamindex_externalreceive[i]]
297                         .leftend_externalheat_GCC, \
298                           streams[streamindex_externalreceive[j]].deltaheat])
299
300
301 #熱容量と実際の外部熱媒使用熱量の計算
302 for i in range (len(streamindex_externalheat)):
303     streams[streamindex_externalheat[i]].deltaheat +=
304               - streams[streamindex_externalheat[i]].leftend_externalheat_GCC
305     streams[streamindex_externalheat[i]].heatcapacity_calc()
306
307 #=============================================================================
308
309 #TQdiagram のデータ制作=======================================================
310
311 sectionheat_give = np.array([0.]*(len(ascendingtemperature_TQ)))
312 sectionheat_receive = np.array([0.]*(len(ascendingtemperature_TQ)))
313 heatsum_give = np.array([0.]*(len(ascendingtemperature_TQ)))
314 heatsum_receive = np.array([0.]*(len(ascendingtemperature_TQ)))
315
316 for i in range (len(ascendingtemperature_TQ)-1):
317     dT = (ascendingtemperature_TQ[i+1] - ascendingtemperature_TQ[i])
318     for j in range (len(streams)):
319         if streams[j].property in gives :
320             if streams[j].temperatureindexMIN \
321                 <= i < streams[j].temperatureindexMAX:
322                 sectionheat_give[i] += streams[j].heatcapacity * dT
323         elif streams[j].property in receives :
324             if streams[j].temperatureindexMIN \
325                 <= i < streams[j].temperatureindexMAX:
326                 sectionheat_receive[i] += streams[j].heatcapacity * dT
```

```
327
328     next
329     heatsum_give[i+1] = sectionheat_give[i] + heatsum_give[i]
330     heatsum_receive[i+1] = sectionheat_receive[i] + heatsum_receive[i]
331 next
332
333 # print("Draw TQ diagram Complete")
334 #=============================================================================
335
336 class HeatExchanger:
337     def __init__(self):
338         self.exchangeheat =0.0
339         self.streamname_give =""
340         self.streamname_receive =""
341         self.temperaturehigh_give =0.0
342         self.temperaturelow_give =0.0
343         self.temperaturehigh_receive =0.0
344         self.temperaturelow_receive =0.0
345         self.heatcapacity_give =0.0
346         self.heatcapacity_receive =0.0
347         self.deltatemperature_high =0.0
348         self.deltatemperature_low =0.0
349         self.LogMeantemperature =0.0
350         self.phase_give =""
351         self.phase_receive =""
352         self.U =0.0
353         self.area =0.0
354         self.rebconfacter =1.
355         self.cost =0.0
356
357
358     def heatcapacity_calc(self):
359         self.heatcapacity_give = self.exchangeheat /
360             (self.temperaturehigh_give - self.temperaturelow_give)
361         self.heatcapacity_receive = self.exchangeheat /
362             (self.temperaturehigh_receive - self.temperaturelow_receive)
363
364     def deltatemperature_LogMean_calc(self):
365         self.deltatemperature_high = np.abs(self.temperaturehigh_give - \
366                                 self.temperaturehigh_receive)
367         self.deltatemperature_low = np.abs(self.temperaturelow_give - \
368                                 self.temperaturelow_receive)
369         if self.deltatemperature_high == self.deltatemperature_low:
370             self.LogMeantemperature = self.deltatemperature_high
371         else:
372             self.LogMeantemperature = (self.deltatemperature_high - \
373                   self.deltatemperature_low)/ \
374                   log(abs(self.deltatemperature_high/self.deltatemperature_low))
```

```
375
376
377     def Rounddata(self):
378         self.exchangeheat = np.round(self.exchangeheat,sig)
379         self.LogMeantemperature = np.round(self.LogMeantemperature,sig)
380         self.heatcapacity_give = np.round(self.heatcapacity_give)
381         self.heatcapacity_receive = np.round(self.heatcapacity_receive)
382
383
384 #複合線図を分解して熱交換器の対応を考える
385 #TQ 関数の作成
386 temperature_GCCgiveinter = interpolate.interp1d(heatsum_give, \
387               ascendingtemperature_TQ ,fill_value='extrapolate', \
388               bounds_error=False)
389 temperature_GCCreceiveinter = interpolate.interp1d(heatsum_receive, \
390               ascendingtemperature_TQ ,fill_value='extrapolate', \
391               bounds_error=False)
392
393 heatsum_give = list(heatsum_give)
394 heatsum_receive = list(heatsum_receive)
395
396 def temperature_TQgive(Q): #引数はascendingtemperature_TQ 参照
397     if Q[1] == give:
398         T = ascendingtemperature_TQ[Q[2]]
399     else:
400         T = temperature_GCCgiveinter(Q[0])
401     return(T)
402
403 def temperature_TQreceive(Q):
404     if Q[1] == receive:
405         T = ascendingtemperature_TQ[Q[2]]
406     else:
407         T = temperature_GCCreceiveinter(Q[0])
408     return(T)
409
410
411 #熱量で区間分割
412 ascendingheat_TQ = []
413 for i in range (len(ascendingtemperature_TQ)):
414     ascendingheat_TQ.append([heatsum_give[i] ,give ,i])
415                     #Q の座標，g,r どちらの区切り点か，どの昇順温度と対応しているかを保存
416     ascendingheat_TQ.append([heatsum_receive[i],receive ,i])
417 next
418 ascendingheat_TQ.sort(key=lambda x:x[0])
419
420 #各区間での熱交換リストの作成
421 streamlistTQsection_give = []
422 streamlistTQsection_receive = []
```

```
423 datalist_heatexchanger = []
424
425 countgive = -1
426 countreceive = -1
427
428 #熱交換データの作成
429 for i in range (len(ascendingheat_TQ)-1):
430
431     if ascendingheat_TQ[i][1] == give: #give ,receive の昇順温度との対応を記録
432         countgive += 1
433     elif ascendingheat_TQ[i][1] == receive:
434         countreceive += 1
435
436     temperaturelow_give = temperature_TQgive(ascendingheat_TQ[i])
437     temperaturelow_receive = temperature_TQreceive(ascendingheat_TQ[i])
438     temperaturehigh_give = temperature_TQgive(ascendingheat_TQ[i+1])
439     temperaturehigh_receive = temperature_TQreceive(ascendingheat_TQ[i+1])
440     deltaT_give = temperaturehigh_give - temperaturelow_give
441     deltaT_receive = temperaturehigh_receive - temperaturelow_receive
442
443     #区間中に存在する熱流体のリストを作成
444     for j in range (len(streams)):
445
446         x = copy.deepcopy(streams[j])
447
448         if x.property in gives:
449             if x.temperatureindexMIN <= countgive < x.temperatureindexMAX:
450                 x.deltaheat_section = x.heatcapacity * deltaT_give
451                 streamlistTQsection_give.append(x)
452         if x.property in receives:
453             if x.temperatureindexMIN <= countreceive < x.temperatureindexMAX:
454                 x.deltaheat_section = x.heatcapacity * deltaT_receive
455                 streamlistTQsection_receive.append(x)
456     next
457
458     #与熱流体と受熱流体のマッチング
459     while len(streamlistTQsection_give) > 0
460         and len(streamlistTQsection_receive) > 0:
461
462         #データクラス作成
463         dummy = HeatExchanger()
464         datalist_heatexchanger.append(dummy)
465
466         heatlist_give_receive = [streamlistTQsection_give[0].deltaheat_section,\
467                                  streamlistTQsection_receive[0].deltaheat_section]
468
469         #熱交換データの構成
470         dlhe = datalist_heatexchanger[-1]
```

```
471         dlhe.exchangeheat = np.min(heatlist_give_receive)
472         dlhe.streamname_give = streamlistTQsection_give[0].name
473         dlhe.streamname_receive = streamlistTQsection_receive[0].name
474         dlhe.temperaturehigh_give = temperaturehigh_give
475         dlhe.temperaturelow_give = temperaturelow_give
476         dlhe.temperaturehigh_receive = temperaturehigh_receive
477         dlhe.temperaturelow_receive = temperaturelow_receive
478         dlhe.heatcapacity_calc()
479         dlhe.phase_give = streamlistTQsection_give[0].phase
480         dlhe.phase_receive = streamlistTQsection_receive[0].phase
481 
482         if streamlistTQsection_give[0].rebcon == "reb"
483             or streamlistTQsection_receive[0].rebcon == "reb":
484             dlhe.rebcon_facter = 2.
485 
486         #熱交換反映
487         streamlistTQsection_give[0].deltaheat_section += - dlhe.exchangeheat
488         streamlistTQsection_receive[0].deltaheat_section += - dlhe.exchangeheat
489 
490         #交換済流体の削除
491         deleteindex = heatlist_give_receive.index(min(heatlist_give_receive))
492         if deleteindex == 0:
493             del streamlistTQsection_give[0]
494         else:
495             del streamlistTQsection_receive[0]
496 next
497 
498 deletecounter = 0 #無視小交換を取り除く処理
499 for i in range(len(datalist_heatexchanger)):
500     datalist_heatexchanger[i].Rounddata()
501     if datalist_heatexchanger[i].exchangeheat < eps:
502         datalist_heatexchanger.insert(0, datalist_heatexchanger[i])
503         del datalist_heatexchanger[i+1]
504         deletecounter += 1
505 for i in range(deletecounter):
506     del datalist_heatexchanger[0]
507 
508 
509 #熱交換データの整理
510 sorted_datalist_heatexchanger =[]
511 datanumberbeforesort = len(datalist_heatexchanger)
512 while len(datalist_heatexchanger) > 0:
513     mixcounter = 1
514     sorted_datalist_heatexchanger
515             .append(copy.deepcopy(datalist_heatexchanger[0]))
516     for i in range(len(datalist_heatexchanger) -1):
517         #交換流体と各熱容量が一致したら熱交換器をまとめる
518         if datalist_heatexchanger[i+1].streamname_give \
```

```
            == datalist_heatexchanger[0].streamname_give \
            and datalist_heatexchanger[i+1].streamname_receive \
            == datalist_heatexchanger[0].streamname_receive \
            and np.abs(datalist_heatexchanger[i+1].heatcapacity_give \
                - datalist_heatexchanger[0].heatcapacity_give) < eps \
            and np.abs(datalist_heatexchanger[i+1].heatcapacity_receive \
                - datalist_heatexchanger[0].heatcapacity_receive) < eps :

            sorted_datalist_heatexchanger[-1].exchangeheat
                += datalist_heatexchanger[i+1].exchangeheat #熱量の加算
            sorted_datalist_heatexchanger[-1].temperaturehigh_give
                = datalist_heatexchanger[i+1].temperaturehigh_give #温度範囲の拡張
            sorted_datalist_heatexchanger[-1].temperaturehigh_receive
                = datalist_heatexchanger[i+1].temperaturehigh_receive

            datalist_heatexchanger.insert(0, datalist_heatexchanger[i+1])
            del datalist_heatexchanger[i+2]
            mixcounter += 1
    next
    for i in range(mixcounter):
        del datalist_heatexchanger[0]
    next
datanumberaftersort = len(sorted_datalist_heatexchanger)

totalcost=0.0
totalheat=0.0

#対数平均温度差dT, 総括伝熱係数 U, 熱交換器面積 A の計算
for i in range(len(sorted_datalist_heatexchanger)):

    sorted_datalist_heatexchanger[i].deltatemperature_LogMean_calc()
    sorted_datalist_heatexchanger[i].Rounddata()

    for j in range (len(twophaselist)):
        if [sorted_datalist_heatexchanger[i].phase_give, \
            sorted_datalist_heatexchanger[i].phase_receive] == twophaselist[j] \
            or [sorted_datalist_heatexchanger[i].phase_receive , \
            sorted_datalist_heatexchanger[i].phase_give] == twophaselist[j]:
            sorted_datalist_heatexchanger[i].U = Ulist[j]
    next

    sorted_datalist_heatexchanger[i].area \
          = (1.e+6/3600.)*sorted_datalist_heatexchanger[i].exchangeheat \
          /(sorted_datalist_heatexchanger[i].U * \
            sorted_datalist_heatexchanger[i].LogMeantemperature)

    sorted_datalist_heatexchanger[i].cost = 0.015 * \
            sorted_datalist_heatexchanger[i].rebconfacter * \
```

```
            sorted_datalist_heatexchanger[i].area ** 0.62
    totalcost += sorted_datalist_heatexchanger[i].cost
    totalheat += sorted_datalist_heatexchanger[i].exchangeheat
next

#出力
wb.sheets[writesheet].clear_contents()
for i in range (len(streams)):
    wb.sheets[readsheet].range((row_readstart + streams[i].number + 1, \
                column_readstart + 2 )).value = streams[i].property
    wb.sheets[readsheet].range((row_readstart + streams[i].number + 1, \
                column_readstart + 3 )).value = streams[i].deltaheat

wb.sheets[writesheet].range((row_writestart, column_writestart +0 )).value
      = "熱交換器番号"
wb.sheets[writesheet].range((row_writestart, column_writestart +1 )).value
      = "与熱流体"
wb.sheets[writesheet].range((row_writestart, column_writestart +2 )).value
      = "受熱流体"
wb.sheets[writesheet].range((row_writestart, column_writestart +3 )).value
      = "対数平均温度差 [K]"
wb.sheets[writesheet].range((row_writestart, column_writestart +4 )).value
      = "交換熱量 [MJ/h]"
wb.sheets[writesheet].range((row_writestart, column_writestart +5 )).value
      = "伝熱面積 [m2]"
wb.sheets[writesheet].range((row_writestart, column_writestart +6 )).value
      = "価格 [億円]"
wb.sheets[writesheet].range((row_writestart, column_writestart +7 )).value
      = "総コスト [億円/year]"
wb.sheets[writesheet].range((row_writestart, column_writestart +8 )).value
      = "総交換熱量 [MJ/h]"
wb.sheets[writesheet].range((row_writestart, column_writestart +10 )).value
      = "温度差 1"
wb.sheets[writesheet].range((row_writestart, column_writestart +11 )).value
      = "温度差 2"

wb.sheets[writesheet].range((row_writestart+1 ,column_writestart +7 )).value
    = totalcost/7.
wb.sheets[writesheet].range((row_writestart+1 ,column_writestart +8 )).value
    = totalheat

for i in range (len(sorted_datalist_heatexchanger)):
    wb.sheets[writesheet].range((row_writestart +i+1 ,column_writestart +0 ))
              .value = i+1
    wb.sheets[writesheet].range((row_writestart +i+1 ,column_writestart +1 ))
              .value = sorted_datalist_heatexchanger[i].streamname_give
    wb.sheets[writesheet].range((row_writestart +i+1 ,column_writestart +2 ))
              .value = sorted_datalist_heatexchanger[i].streamname_receive
```

```
    wb.sheets[writesheet].range((row_writestart +i+1 ,column_writestart +3 ))
              .value = sorted_datalist_heatexchanger[i].LogMeantemperature
    wb.sheets[writesheet].range((row_writestart +i+1 ,column_writestart +4 ))
              .value = sorted_datalist_heatexchanger[i].exchangeheat
    wb.sheets[writesheet].range((row_writestart +i+1 ,column_writestart +5 ))
              .value = sorted_datalist_heatexchanger[i].area
    wb.sheets[writesheet].range((row_writestart +i+1 ,column_writestart +6 ))
              .value = sorted_datalist_heatexchanger[i].cost

    wb.sheets[writesheet].range((row_writestart +i+1 ,column_writestart+10 ))
              .value = sorted_datalist_heatexchanger[i].deltatemperature_high
    wb.sheets[writesheet].range((row_writestart +i+1 ,column_writestart +11 ))
              .value = sorted_datalist_heatexchanger[i].deltatemperature_low
next


#============================================================================
# #グランドコンポジットカーブのデータ出力

rcParams['font.family'] = 'sans-serif'
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic', \
                            'Meirio', 'Takao', 'IPAexGothic', 'IPAPGothic', \
                            'VL PGothic', 'Noto Sans CJK JP']
rcParams['font.size'] = 22

# fig1=plt.figure()
# ax1 = fig1.add_subplot()
# ax1.plot(heatbalance_sum_GCC, \
#    ascendingtemperature_GCC,label="グランドコンポジットカーブ")
# # plt.xlim(0,40)
# # plt.ylim(0,250)
# # ax1.set_xlabel(xlabel="熱量 [MJ/h]")
# # ax1.set_ylabel(ylabel="温度 [℃]")
# plt.gca().xaxis.set_tick_params(which='both',direction='in', \
#      bottom=True ,top=True ,left=True ,right=True)
# plt.gca().yaxis.set_tick_params(which='both',direction='in', \
#      bottom=True ,top=True ,left=True ,right=True)
# # plt.rcParams["legend.fancybox"] = False # 丸角
# # plt.rcParams["legend.framealpha"] = 1 # 透明度の指定，0で塗りつぶしなし
# # plt.rcParams["legend.edgecolor"] = 'black' # edge の色を変更
# # plt.legend()
# plt.show()



# print("Draw GCC diagram Complete")

#TQ 線図出力
```

```
fig2 = plt.figure()
ax_give = fig2.add_subplot()
ax_receive = fig2.add_subplot()
heatsum_give = list(heatsum_give)
heatsum_receive = list(heatsum_receive)
ascendingtemperature_TQ = list(ascendingtemperature_TQ)

ax_give.plot(heatsum_give, ascendingtemperature_TQ ,color ="r")
ax_receive.plot(heatsum_receive, ascendingtemperature_TQ ,color ="b")
plt.xlim(0,heatsum_receive[-1])
plt.ylim(-10,ascendingtemperature_TQ[-1]+10)
# ax_give.set_xlabel(xlabel="熱量 [MJ/h]")
# ax_give.set_ylabel(ylabel="温度 [℃]")
plt.gca().xaxis.set_tick_params(which='both',direction='in',bottom=True ,\
      top=True ,left=True ,right=True)
plt.gca().yaxis.set_tick_params(which='both',direction='in',bottom=True ,\
      top=True ,left=True ,right=True)
# plt.rcParams["legend.fancybox"] = False # 丸角
# plt.rcParams["legend.framealpha"] = 1 # 透明度の指定，0で塗りつぶしなし
# plt.rcParams["legend.edgecolor"] = 'black' # edge の色を変更
# plt.legend()
plt.tick_params(axis = 'x', colors ='k')
plt.tick_params(axis = 'y', colors = 'k')
plt.show()
```

B.2 VBA

ソースコード B.4 変数宣言

```
Option Explicit

Public Const TolMolarMass = 92.141 'トルエンモル質量 [kg/kmol]
Public Const BzAMolarMass = 122.124 '安息香酸モル質量 [kg/kmol]
Public Const WaterMolarMass = 18.015 '水モル質量 [kg/kmol]

Public Const OxygenFraction = 0.2
Public Const NytrogenFraction = 0.8

Public hycase As Object

Public AdditionalAirHYSYS As ProcessStream
Public AdopterHYSYS As ProcessStream
Public BoilerBottomOutHYSYS As ProcessStream
Public BoilerTopOutHYSYS As ProcessStream
Public CrystallizationInHYSYS As ProcessStream
Public CrystallizedRecycleHYSYS As ProcessStream
Public DecanterOutHYSYS As ProcessStream
Public DistillationInHYSYS As ProcessStream
```

```
Public DistillationLiqOutHYSYS As ProcessStream
Public DistillationVapOutHYSYS As ProcessStream
Public DistVapOutXHYSYS As ProcessStream
Public ExternalOutHYSYS As ProcessStream
Public ExtractedBzAHYSYS As ProcessStream
Public ExtractedOthersHYSYS As ProcessStream
Public ExtractedWaterHYSYS As ProcessStream
Public FeedHYSYS As ProcessStream
Public MixerInHYSYS As ProcessStream
Public PostDistillationVapOutHYSYS As ProcessStream
Public PostDistVapOutXHYSYS As ProcessStream
Public PostReactorOutHYSYS As ProcessStream
Public PreAdopterHYSYS As ProcessStream
Public PreCrystallizedRecycleHYSYS As ProcessStream
Public PreReactorInHYSYS As ProcessStream
Public ProductHYSYS As ProcessStream
Public PumpInHYSYS As ProcessStream
Public ReactorInHYSYS As ProcessStream
Public ReactorOutHYSYS As ProcessStream
Public ValveOutHYSYS As ProcessStream

Public PreReactorInHeaterHYSYS As ProcessStream
Public ReactorOutPumpHYSYS As ProcessStream
Public CondenserHYSYS As ProcessStream
Public ReboilerHYSYS As ProcessStream
Public ExternalOutCoolerHYSYS As ProcessStream
Public PreCrystallizedRecycleHeaterHYSYS As ProcessStream
Public ValveOutCoolerHYSYS As ProcessStream
Public PowerOfPumpHYSYS As ProcessStream
Public DistVapOutCoolerHYSYS As ProcessStream
Public DistVapOutExpandHYSYS As ProcessStream
```

ソースコード B.5　HYSYS と python を繋ぐコード

```
Option Explicit

Sub main()

Dim topRow As Integer
topRow = 18

Dim resultSheet As Worksheet
Set resultSheet = ThisWorkbook.Sheets(1)

With resultSheet
    .Cells(10, 4) = ""
    .Cells(11, 4) = ""
    .Range(.Cells(18, 4), .Cells(48, 12)) = ""
    .Range(.Cells(18, 11), .Cells(50, 12)) = 0
```

```
End With


Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer
Dim h As Integer
Dim p As Integer
Dim q As Integer

Dim flowRange As Variant 'excel シートから array への adopter
Dim reactorErr As Double '反応器収束用誤差
Dim reactorEps As Double '反応器収束用
Dim crystallizerErr As Double '晶析器収束用誤差
Dim crystallizerEps As Double '晶析器収束用

reactorErr = 1
crystallizerErr = 1
reactorEps = 0.001
crystallizerEps = 0.001

'-------------------------------------------------------------------------------
Dim k As Variant '[h-1] 反応速度定数
Dim reactorPres As Double '[bar] 反応器圧力
Dim reactorVol As Double '[m3] 反応器体積
Dim reactorTemp As Double '[℃] 反応器温度
Dim F0 As Double '[kmol/h] feed トルエン流量
Dim extractorTemp As Double '[℃] 抽出温度
Dim crystallizerVolume As Double '[℃] 晶析器体積
Dim crystallizerTemp As Double '[℃] 晶析器温度

With resultSheet
    reactorPres = .Cells(3, 4)
    reactorVol = .Cells(4, 4)
    reactorTemp = .Cells(5, 4) + 273.15
    F0 = .Cells(6, 4)
    extractorTemp = 40
    crystallizerVolume = .Cells(8, 4)
    crystallizerTemp = .Cells(9, 4)
End With
Call Utils.solve_k(reactorTemp, k)
'-------------------------------------------------------------------------------

'-------------------------------------------------------------------------------
Set hycase = GetObject(ThisWorkbook.Path & "¥main.hsc")

With hycase.Flowsheet.MaterialStreams
```

```
64      Set AdditionalAirHYSYS = .Item("AdditionalAir")
65      Set AdopterHYSYS = .Item("Adopter")
66      Set BoilerBottomOutHYSYS = .Item("BoilerBottomOut")
67      Set BoilerTopOutHYSYS = .Item("BoilerTopOut")
68      Set CrystallizationInHYSYS = .Item("CrystallizationIn")
69      Set CrystallizedRecycleHYSYS = .Item("CrystallizedRecycle")
70      Set DecanterOutHYSYS = .Item("DecanterOut")
71      Set DistillationInHYSYS = .Item("DistillationIn")
72      Set DistillationLiqOutHYSYS = .Item("DistillationLiqOut")
73      Set DistillationVapOutHYSYS = .Item("DistillationVapOut")
74      Set DistVapOutXHYSYS = .Item("DistVapOutX")
75      Set ExternalOutHYSYS = .Item("ExternalOut")
76      Set ExtractedBzAHYSYS = .Item("ExtractedBzA")
77      Set ExtractedOthersHYSYS = .Item("ExtractedOthers")
78      Set ExtractedWaterHYSYS = .Item("ExtractedWater")
79      Set FeedHYSYS = .Item("Feed")
80      Set MixerInHYSYS = .Item("MixerIn")
81      Set PostDistillationVapOutHYSYS = .Item("PostDistillationVapOut")
82      Set PostDistVapOutXHYSYS = .Item("PostDistVapOutX")
83      Set PostReactorOutHYSYS = .Item("PostReactorOut")
84      Set PreAdopterHYSYS = .Item("PreAdopter")
85      Set PreCrystallizedRecycleHYSYS = .Item("PreCrystallizedRecycle")
86      Set PreReactorInHYSYS = .Item("PreReactorIn")
87      Set ProductHYSYS = .Item("Product")
88      Set PumpInHYSYS = .Item("PumpIn")
89      Set ReactorInHYSYS = .Item("ReactorIn")
90      Set ReactorOutHYSYS = .Item("ReactorOut")
91      Set ValveOutHYSYS = .Item("ValveOut")
92  End With
93
94  With hycase.Flowsheet.EnergyStreams
95      Set PreReactorInHeaterHYSYS = .Item("PreReactorInHeater")
96      Set ReactorOutPumpHYSYS = .Item("ReactorOutPump")
97      Set CondenserHYSYS = .Item("Condenser")
98      Set ReboilerHYSYS = .Item("Reboiler")
99      Set ExternalOutCoolerHYSYS = .Item("ExternalOutCooler")
100     Set PreCrystallizedRecycleHeaterHYSYS = _
101         .Item("PreCrystallizedRecycleHeater")
102     Set ValveOutCoolerHYSYS = .Item("ValveOutCooler")
103     Set PowerOfPumpHYSYS = .Item("PowerOfPump")
104     Set DistVapOutCoolerHYSYS = .Item("DistVapOutCooler")
105     Set DistVapOutExpandHYSYS = .Item("DistVapOutExpand")
106 End With
107
108 Dim DistillationHYSYS As ColumnOp
109
110 Dim ToReb As ProcessStream
111 Dim BoilUp As ProcessStream
```

```
Dim ToCond As ProcessStream
Dim Reflux As ProcessStream

Set DistillationHYSYS = hycase.Flowsheet.Operations.Item("T-100")
With DistillationHYSYS.ColumnFlowsheet.MaterialStreams
    Set ToReb = .Item("To Reboiler")
    Set BoilUp = .Item("Boilup")
    Set ToCond = .Item("To Condenser")
    Set Reflux = .Item("Reflux")
End With
'--------------------------------------------------------------------------------

'--------------------------------------------------------------------------------
With resultSheet
    AdditionalAirHYSYS.Pressure.SetValue .Cells(topRow + 0, 1), "bar"
    AdopterHYSYS.Pressure.SetValue .Cells(topRow + 1, 1), "bar"
    CrystallizedRecycleHYSYS.Pressure.SetValue .Cells(topRow + 5, 1), "bar"
    DecanterOutHYSYS.Pressure.SetValue .Cells(topRow + 6, 1), "bar"
    DistillationInHYSYS.Pressure.SetValue .Cells(topRow + 7, 1), "bar"
    DistVapOutXHYSYS.Pressure.SetValue .Cells(topRow + 10, 1), "bar"
    ExternalOutHYSYS.Pressure.SetValue .Cells(topRow + 11, 1), "bar"
    ExtractedBzAHYSYS.Pressure.SetValue .Cells(topRow + 12, 1), "bar"
    ExtractedOthersHYSYS.Pressure.SetValue .Cells(topRow + 13, 1), "bar"
    ExtractedWaterHYSYS.Pressure.SetValue .Cells(topRow + 14, 1), "bar"
    FeedHYSYS.Pressure.SetValue .Cells(topRow + 15, 1), "bar"
    PostDistillationVapOutHYSYS.Pressure.SetValue .Cells(topRow + 17, 1), "bar"
    PostDistVapOutXHYSYS.Pressure.SetValue .Cells(topRow + 18, 1), "bar"
    PostReactorOutHYSYS.Pressure.SetValue .Cells(topRow + 19, 1), "bar"
    PreAdopterHYSYS.Pressure.SetValue .Cells(topRow + 20, 1), "bar"
    PreCrystallizedRecycleHYSYS.Pressure.SetValue .Cells(topRow + 21, 1), "bar"
    PreReactorInHYSYS.Pressure.SetValue .Cells(topRow + 22, 1), "bar"
    ProductHYSYS.Pressure.SetValue .Cells(topRow + 23, 1), "bar"
    ReactorInHYSYS.Pressure.SetValue .Cells(topRow + 25, 1), "bar"
    ReactorOutHYSYS.Pressure.SetValue .Cells(topRow + 26, 1), "bar"
    ValveOutHYSYS.Pressure.SetValue .Cells(topRow + 27, 1), "bar"


    AdditionalAirHYSYS.Temperature.SetValue .Cells(topRow + 0, 2), "C"
    AdopterHYSYS.Temperature.SetValue .Cells(topRow + 1, 2), "C"
    CrystallizedRecycleHYSYS.Temperature.SetValue .Cells(topRow + 5, 2), "C"
    DecanterOutHYSYS.Temperature.SetValue .Cells(topRow + 6, 2), "C"
    ExternalOutHYSYS.Temperature.SetValue .Cells(topRow + 11, 2), "C"
    ExtractedBzAHYSYS.Temperature.SetValue .Cells(topRow + 12, 2), "C"
    ExtractedOthersHYSYS.Temperature.SetValue .Cells(topRow + 13, 2), "C"
    ExtractedWaterHYSYS.Temperature.SetValue .Cells(topRow + 14, 2), "C"
    FeedHYSYS.Temperature.SetValue .Cells(topRow + 15, 2), "C"
    PreAdopterHYSYS.Temperature.SetValue .Cells(topRow + 20, 2), "C"
    PreCrystallizedRecycleHYSYS.Temperature.SetValue .Cells(topRow + 21, 2), "C"
```

```
    ProductHYSYS.Temperature.SetValue .Cells(topRow + 23, 2), "C"
    ReactorInHYSYS.Temperature.SetValue .Cells(topRow + 25, 2), "C"
    ReactorOutHYSYS.Temperature.SetValue .Cells(topRow + 26, 2), "C"
End With
'-------------------------------------------------------------------------------

'-------------------------------------------------------------------------------
Dim feedMolarFlow As Double 'feed モル流量 [kmol/h]
Dim feedVolumeFlow As Double 'feed 体積流量 [m3/h]
Dim feedComponentMolarFlow As Variant 'feed 成分別モル流量 [kmol/h]
Dim feedComponentVolumeFlow As Variant 'feed 成分別体積流量 [m3/h]
Dim feedComponentMolarFractin As Variant 'feed モル分率 [-]

Dim reactorInMolarFlow As Double 'CSTR 入口モル流量 [kmol/h]
Dim reactorInVolumeFlow As Double 'CSTR 入口体積流量 [m3/h]
Dim reactorInComponentMolarFlow As Variant 'CSTR 入口成分別モル流量 [kmol/h]
Dim reactorInComponentVolumeFlow As Variant 'CSTR 入口成分別体積流量 [m3/h]
Dim reactorInComponentMolarFraction As Variant 'CSTR 入口モル分率 [-]

Dim reactorOutMolarFlow As Double 'CSTR 出口モル流量 [kmol/h]
Dim reactorOutVolumeFlow As Double 'CSTR 出口体積流量 [m3/h]
Dim reactorOutComponentMolarFlow As Variant 'CSTR 出口成分別モル流量 [kmol/h]
Dim reactorOutComponentVolumeFlow As Variant 'CSTR 出口成分別体積流量 [m3/h]
Dim reactorOutComponentMolarFraction As Variant 'CSTR 出口モル分率 [-]

Dim decanterOutComponentMolarFlow As Variant 'デカンタ気体出口成分別モル流量 [kmol/h]

Dim boilerInComponentMolarFlow As Variant 'ボイラー入口成分別モル流量 [kmol/h]
Dim distillationLiqOutComponentMolarFlow As Variant
                                        '蒸留塔缶出液成分別モル流量 [kmol/h]

Dim extractedWaterMolarFlow As Double '抽出器出口水モル流量
Dim extractedWaterComponentMolarFlow As Variant '抽出器出口水成分別モル流量

Dim extractedBzAMolarFlow As Double '抽出器出口安息香酸モル流量
Dim extractedBzAComponentMolarFlow As Variant '抽出器出口安息香酸成分別モル流量

Dim extractedOthersMolarFlow As Double '抽出器出口不純物モル流量
Dim extractedOthersComponentMolarFlow As Variant '抽出器出口不純物成分別モル流量

Dim adopterComponentMolarFlow As Variant 'リサイクル計算用蒸留塔缶出液流量 [kmol/h]

Dim ToCrystallizationMolarFlow As Double '晶析器入口モル流量 [kmol/h]
Dim ToCrystallizationVolumeFlow As Double '晶析器入口体積流量 [m3/h]
Dim ToCrystallizationBzAMolarFraction As Double '晶析器入口安息香酸モル分率 [-]
Dim ToCrystallizationBzAMolarFlow As Double '晶析器入口安息香酸モル流量 [kmol/h]
Dim ToCrystallizationWaterMolarFlow As Double '晶析器入口水モル流量 [kmol/h]

```

```
208 Dim BzAConcentration As Double '晶析器入口安息香酸濃度 [kg/m3]
209 Dim CrystallizedBzA As Double '晶析した安息香酸モル流量 [kmole/h]
210 Dim products As Variant '晶析した安息香酸
211 Dim restOfProducts As Variant '晶析しなかった残り
212 Dim RestOfBzA As Double '晶析しなかった安息香酸
213 Dim RestOfWater As Double '晶析器から流出する水
214
215 Dim necessaryOxygen As Double 'トルエン完全燃焼に必要な酸素 [kmol/h]
216 Dim lackingOxygen As Double '不足している酸素 [kmol/h]
217 Dim additionalOxygen As Double '追加する酸素 [kmol/h]
218 Dim additionalNytrogen As Double '追加する窒素 [kmol/h]
219 Dim additionalAir As Variant '追加する空気 [kmol/h]
220 Dim isAirNeed As Boolean 'whether additional air is needed
221 isAirNeed = False
222
223 Dim distillationVapOutTmp As Double '蒸留塔留出液の温度
224 Dim distillationVapOutComponentMolarFlow As Variant
225                                     '蒸留塔留出液の成分別モル流量 [kmol/h]
226 '-------------------------------------------------------------------------------
227
228 '-------------------------------------------------------------------------------
229 feedComponentMolarFlow = Array(F0, 0, 0, 0, 0, 0, 0, 0, 0)
230 feedComponentVolumeFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
231 feedComponentMolarFractin = Array(1, 0, 0, 0, 0, 0, 0, 0, 0)
232
233 reactorInComponentMolarFlow = Array(F0, 0, 0, 0, 0, 0, 0, 0, 0)
234 reactorInComponentVolumeFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
235 reactorInComponentMolarFraction = Array(1, 0, 0, 0, 0, 0, 0, 0, 0)
236 Call Utils.solveComponentVolumeFlow(reactorInComponentMolarFlow, _
237               reactorInComponentVolumeFlow)
238 reactorInMolarFlow = Utils.sumOfFlow(reactorInComponentMolarFlow)
239 reactorInVolumeFlow = Utils.sumOfFlow(reactorInComponentVolumeFlow)
240
241 reactorOutComponentMolarFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
242 reactorOutComponentVolumeFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
243 reactorOutComponentMolarFraction = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
244
245
246 boilerInComponentMolarFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
247 decanterOutComponentMolarFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
248
249 distillationLiqOutComponentMolarFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
250
251 extractedBzAComponentMolarFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
252 extractedOthersComponentMolarFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
253 extractedWaterComponentMolarFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
254
255 adopterComponentMolarFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
```

```
256
257 distillationVapOutComponentMolarFlow = Array(0, 0, 0, 0, 0, 0, 0, 0, 0)
258 '-------------------------------------------------------------------------------
259
260
261 '-------------------------------------------------------------------------------
262 FeedHYSYS.ComponentMolarFlow.SetValues feedComponentMolarFlow, "kgmole/h"
263 i = 0
264 Do While (reactorErr > reactorEps And i < 600)
265     With resultSheet
266         .Range(.Cells(topRow + 25, 4), .Cells(topRow + 25, 12)) = _
267               reactorInComponentMolarFlow
268         .Cells(topRow + 25, 14) = reactorInVolumeFlow
269         ' 単通反応率
270         .Cells(12, 4) = (reactorInComponentMolarFlow(0) - _
271                         reactorOutComponentMolarFlow(0)) / _
272                         reactorInComponentMolarFlow(0)
273     End With
274
275     Call RunPython("import analyzer_nowrite; analyzer_nowrite.main()")
276
277     ' 反応器出口のモル流量をシートから取得する.
278     resultSheet.Cells(topRow + 26, 9) = 0
279     resultSheet.Cells(topRow + 26, 10) = 0
280     flowRange = resultSheet.Range(resultSheet.Cells(topRow + 26, 4), _
281                 resultSheet.Cells(topRow + 26, 12))
282     For j = 1 To 9
283         reactorOutComponentMolarFlow(j - 1) = flowRange(1, j)
284     Next j
285     reactorOutComponentMolarFlow(5) = 0
286     reactorOutComponentMolarFlow(6) = 0
287
288     ' デカンタから蒸発するモル流量をシートから取得する.
289     flowRange = resultSheet.Range(resultSheet.Cells(topRow + 31, 4), _
290                 resultSheet.Cells(topRow + 31, 12))
291     For j = 1 To 9
292         decanterOutComponentMolarFlow(j - 1) = flowRange(1, j)
293     Next j
294
295     ' HYSYS に反応器出口とデカンタ出口のモル流量を渡す.
296     ReactorOutHYSYS.ComponentMolarFlow.SetValues _
297                 reactorOutComponentMolarFlow, "kgmole/h"
298     DecanterOutHYSYS.ComponentMolarFlow.SetValues _
299                 decanterOutComponentMolarFlow, "kgmole/h"
300
301     ' 蒸留塔留出液のつじつま合わせ.
302     distillationVapOutTmp = PostDistillationVapOutHYSYS. _
303                   Temperature.GetValue("C")
```

```
304     distillationVapOutComponentMolarFlow = PostDistillationVapOutHYSYS. _
305                  ComponentMolarFlow.GetValues("kgmole/h")
306     distillationVapOutComponentMolarFlow(5) = 0
307     distillationVapOutComponentMolarFlow(6) = 0
308
309     DistVapOutXHYSYS.Temperature.SetValue distillationVapOutTmp, "C"
310     DistVapOutXHYSYS.ComponentMolarFlow.SetValues _
311                distillationVapOutComponentMolarFlow, "kgmole/h"
312
313
314     ' 蒸留塔缶出液のモル流量をHYSYS から取得し，疑似抽出を行う.
315     distillationLiqOutComponentMolarFlow = DistillationLiqOutHYSYS. _
316                ComponentMolarFlow.GetValues("kgmole/h")
317
318     Call pseudo_extractor.main(distillationLiqOutComponentMolarFlow, _
319                                extractedBzAComponentMolarFlow, _
320                                extractedOthersComponentMolarFlow, _
321                                extractedWaterComponentMolarFlow, _
322                                extractorTemp)
323
324     ExtractedBzAHYSYS.ComponentMolarFlow.SetValues _
325              extractedBzAComponentMolarFlow, "kgmole/h"
326     ExtractedOthersHYSYS.ComponentMolarFlow.SetValues _
327              extractedOthersComponentMolarFlow, "kgmole/h"
328     ExtractedWaterHYSYS.ComponentMolarFlow.SetValues _
329              extractedWaterComponentMolarFlow, "kgmole/h"
330
331     ' 疑似抽出の結果を晶析器に渡す.
332     ToCrystallizationBzAMolarFlow = extractedBzAComponentMolarFlow(3)
333     ToCrystallizationWaterMolarFlow = extractedWaterComponentMolarFlow(4)
334     ToCrystallizationVolumeFlow = CrystallizationInHYSYS. _
335                                   StdLiqVolFlow.GetValue("m3/h")
336
337     BzAConcentration = (ToCrystallizationBzAMolarFlow * BzAMolarMass) / _
338                        (ToCrystallizationWaterMolarFlow * WaterMolarMass)
339
340     CrystallizedBzA = crystallization.MSMPR(crystallizerTemp, _
341                                             BzAConcentration, _
342                                             ToCrystallizationVolumeFlow, _
343                                             crystallizerVolume, _
344                                             ToCrystallizationBzAMolarFlow)
345
346     RestOfBzA = ToCrystallizationBzAMolarFlow - CrystallizedBzA
347     RestOfWater = ToCrystallizationWaterMolarFlow
348     products = Array(0, 0, 0, CrystallizedBzA, 0, 0, 0, 0, 0)
349     restOfProducts = Array(0, 0, 0, RestOfBzA, RestOfWater, 0, 0, 0, 0)
350     ProductHYSYS.ComponentMolarFlow.SetValues products, "kgmole/h"
351     PreCrystallizedRecycleHYSYS.ComponentMolarFlow.SetValues _
```

```
352              restOfProducts, "kgmole/h"
353
354     ' 収束判定を行う.
355     reactorErr = Abs((distillationLiqOutComponentMolarFlow(3) + _
356                       decanterOutComponentMolarFlow(0) - F0)) / F0
357
358     reactorInComponentMolarFlow = ReactorInHYSYS. _
359                     ComponentMolarFlow.GetValues("kgmole/h")
360     reactorInVolumeFlow = ReactorInHYSYS.StdLiqVolFlow.GetValue("m3/h")
361
362
363     i = i + 1
364 Loop
365
366 '----------反応器収束後,シートに計算結果を書き込むバージョンのpython を実行する. --------
367 Call RunPython("import analyzer_final; analyzer_final.main()")
368 resultSheet.Cells(topRow + 26, 9) = 0
369 resultSheet.Cells(topRow + 26, 10) = 0
370 flowRange = resultSheet.Range(resultSheet.Cells(topRow + 26, 4), _
371             resultSheet.Cells(topRow + 26, 12))
372 For j = 1 To 9
373     reactorOutComponentMolarFlow(j - 1) = flowRange(1, j)
374 Next j
375 flowRange = resultSheet.Range(resultSheet.Cells(topRow + 31, 4), _
376             resultSheet.Cells(topRow + 31, 12))
377 For j = 1 To 9
378     decanterOutComponentMolarFlow(j - 1) = flowRange(1, j)
379 Next j
380 reactorOutComponentMolarFlow(8) = 0
381 ReactorOutHYSYS.ComponentMolarFlow.SetValues _
382           reactorOutComponentMolarFlow, "kgmole/h"
383 DecanterOutHYSYS.ComponentMolarFlow.SetValues _
384           decanterOutComponentMolarFlow, "kgmole/h"
385
386 distillationLiqOutComponentMolarFlow = _
387           DistillationLiqOutHYSYS.ComponentMolarFlow.GetValues("kgmole/h")
388
389 Call pseudo_extractor.main(distillationLiqOutComponentMolarFlow, _
390                            extractedBzAComponentMolarFlow, _
391                            extractedOthersComponentMolarFlow, _
392                            extractedWaterComponentMolarFlow, _
393                            extractorTemp)
394
395 ExtractedWaterHYSYS.ComponentMolarFlow.SetValues _
396               extractedWaterComponentMolarFlow, "kgmole/h"
397 ExtractedBzAHYSYS.ComponentMolarFlow.SetValues _
398               extractedBzAComponentMolarFlow, "kgmole/h"
399 ExtractedOthersHYSYS.ComponentMolarFlow.SetValues _
```

```
400              extractedOthersComponentMolarFlow, "kgmole/h"
401
402 ToCrystallizationBzAMolarFlow = extractedBzAComponentMolarFlow(3)
403 ToCrystallizationWaterMolarFlow = extractedWaterComponentMolarFlow(4)
404 ToCrystallizationVolumeFlow = CrystallizationInHYSYS. _
405                 StdLiqVolFlow.GetValue("m3/h")
406
407 BzAConcentration = (ToCrystallizationBzAMolarFlow * BzAMolarMass) / _
408                    (ToCrystallizationWaterMolarFlow * WaterMolarMass)
409
410 CrystallizedBzA = crystallization.MSMPR(crystallizerTemp, _
411                                         BzAConcentration, _
412                                         ToCrystallizationVolumeFlow, _
413                                         crystallizerVolume, _
414                                         ToCrystallizationBzAMolarFlow)
415
416 RestOfBzA = ToCrystallizationBzAMolarFlow - CrystallizedBzA
417 RestOfWater = ToCrystallizationWaterMolarFlow
418 products = Array(0, 0, 0, CrystallizedBzA, 0, 0, 0, 0, 0)
419 restOfProducts = Array(0, 0, 0, RestOfBzA, RestOfWater, 0, 0, 0, 0)
420 ProductHYSYS.ComponentMolarFlow.SetValues products, "kgmole/h"
421 PreCrystallizedRecycleHYSYS.ComponentMolarFlow.SetValues _
422                       restOfProducts, "kgmole/h"
423
424 '-------------------------------------------------------------------------
425
426
427
428 '--------------------抽出器の収束計算を行う--------------------
429 ' 蒸留塔出口のモル流量は操作できないので,晶析のリサイクルを行うために,
430 ' Adopter を設置し，そこにで流量を操作する.
431 ' まず,preAdopter の成分別モル流量を設定する.
432 AdopterHYSYS.ComponentMolarFlow.SetValues _
433         PreAdopterHYSYS.ComponentMolarFlow.GetValues("kgmole/h"), "kgmole/h"
434
435 i = 0
436 Do While (crystallizerErr > crystallizerEps And i < 100)
437     adopterComponentMolarFlow = AdopterHYSYS. _
438                 ComponentMolarFlow.GetValues("kgmole/h")
439
440     Call pseudo_extractor.main(adopterComponentMolarFlow, _
441                                extractedBzAComponentMolarFlow, _
442                                extractedOthersComponentMolarFlow, _
443                                extractedWaterComponentMolarFlow, _
444                                extractorTemp)
445
446     ExtractedWaterHYSYS.ComponentMolarFlow.SetValues _
447                   extractedWaterComponentMolarFlow, "kgmole/h"
```

```
448     ExtractedBzAHYSYS.ComponentMolarFlow.SetValues _
449                 extractedBzAComponentMolarFlow, "kgmole/h"
450     ExtractedOthersHYSYS.ComponentMolarFlow.SetValues _
451                 extractedOthersComponentMolarFlow, "kgmole/h"
452
453     ToCrystallizationBzAMolarFlow = extractedBzAComponentMolarFlow(3)
454     ToCrystallizationWaterMolarFlow = extractedWaterComponentMolarFlow(4)
455     ToCrystallizationVolumeFlow = CrystallizationInHYSYS. _
456                 StdLiqVolFlow.GetValue("m3/h")
457
458     BzAConcentration = (ToCrystallizationBzAMolarFlow * BzAMolarMass) / _
459                         (ToCrystallizationWaterMolarFlow * WaterMolarMass)
460
461     CrystallizedBzA = crystallization.MSMPR(crystallizerTemp, _
462                                             BzAConcentration, _
463                                             ToCrystallizationVolumeFlow, _
464                                             crystallizerVolume, _
465                                             ToCrystallizationBzAMolarFlow)
466
467     RestOfBzA = ToCrystallizationBzAMolarFlow - CrystallizedBzA
468     RestOfWater = ToCrystallizationWaterMolarFlow
469     products = Array(0, 0, 0, CrystallizedBzA, 0, 0, 0, 0, 0)
470     restOfProducts = Array(0, 0, 0, RestOfBzA, RestOfWater, 0, 0, 0, 0)
471     ProductHYSYS.ComponentMolarFlow.SetValues products, "kgmole/h"
472     PreCrystallizedRecycleHYSYS.ComponentMolarFlow.SetValues _
473                   restOfProducts, "kgmole/h"
474
475     adopterComponentMolarFlow(3) = distillationLiqOutComponentMolarFlow(3) + _
476                                    restOfProducts(3)
477     AdopterHYSYS.ComponentMolarFlow.SetValues _
478                             adopterComponentMolarFlow, "kgmole/h"
479
480     crystallizerErr = Abs((distillationLiqOutComponentMolarFlow(3) + _
481                       restOfProducts(3)) - (products(3) + restOfProducts(3)))
482     i = i + 1
483 Loop
484 '-------------------------------------------------------------------------------
485
486 '-------------------------------------------------------------------------------
487 decanterOutComponentMolarFlow = _
488         DecanterOutHYSYS.ComponentMolarFlow.GetValues("kgmole/h")
489
490 ' 量論関係から,トルエンを全量燃焼させるのに必要な酸素量を求める.
491 ' Tol + 9 O2 -> 4 H2O + 7 CO2
492 necessaryOxygen = decanterOutComponentMolarFlow(0) * 9
493 lackingOxygen = necessaryOxygen - decanterOutComponentMolarFlow(6)
494 additionalOxygen = lackingOxygen
495 additionalNytrogen = lackingOxygen * (NytrogenFraction / OxygenFraction)
```

```
496
497
498 If (additionalOxygen < 0) Then
499     additionalOxygen = 0
500     additionalNytrogen = 0
501     isAirNeed = True
502 End If
503
504 additionalAir = Array(0, 0, 0, 0, 0, additionalNytrogen, additionalOxygen, 0, 0)
505 AdditionalAirHYSYS.ComponentMolarFlow.SetValues additionalAir, "kgmole/h"
506 '-------------------------------------------------------------------------------
507
508 '-------------------------------------------------------------------------------
509 With resultSheet
510     For i = 0 To 8
511         .Cells(topRow + 0, i + 4) = _
512               AdditionalAirHYSYS.ComponentMolarFlow(i) * 3600
513         .Cells(topRow + 1, i + 4) = _
514               AdopterHYSYS.ComponentMolarFlow(i) * 3600
515         .Cells(topRow + 2, i + 4) = _
516               BoilerBottomOutHYSYS.ComponentMolarFlow(i) * 3600
517         .Cells(topRow + 3, i + 4) = _
518               BoilerTopOutHYSYS.ComponentMolarFlow(i) * 3600
519         .Cells(topRow + 4, i + 4) = _
520               CrystallizationInHYSYS.ComponentMolarFlow(i) * 3600
521         .Cells(topRow + 5, i + 4) = _
522               CrystallizedRecycleHYSYS.ComponentMolarFlow(i) * 3600
523         .Cells(topRow + 6, i + 4) = _
524               DecanterOutHYSYS.ComponentMolarFlow(i) * 3600
525         .Cells(topRow + 7, i + 4) = _
526               DistillationInHYSYS.ComponentMolarFlow(i) * 3600
527         .Cells(topRow + 8, i + 4) = _
528               DistillationLiqOutHYSYS.ComponentMolarFlow(i) * 3600
529         .Cells(topRow + 9, i + 4) = _
530               DistillationVapOutHYSYS.ComponentMolarFlow(i) * 3600
531         .Cells(topRow + 10, i + 4) = _
532               DistVapOutXHYSYS.ComponentMolarFlow(i) * 3600
533         .Cells(topRow + 11, i + 4) = _
534               ExternalOutHYSYS.ComponentMolarFlow(i) * 3600
535         .Cells(topRow + 12, i + 4) = _
536               ExtractedBzAHYSYS.ComponentMolarFlow(i) * 3600
537         .Cells(topRow + 13, i + 4) = _
538               ExtractedOthersHYSYS.ComponentMolarFlow(i) * 3600
539         .Cells(topRow + 14, i + 4) = _
540               ExtractedWaterHYSYS.ComponentMolarFlow(i) * 3600
541         .Cells(topRow + 15, i + 4) = _
542               FeedHYSYS.ComponentMolarFlow(i) * 3600
543         .Cells(topRow + 16, i + 4) = _
```

```
544                 MixerInHYSYS.ComponentMolarFlow(i) * 3600
545             .Cells(topRow + 17, i + 4) = _
546                 PostDistillationVapOutHYSYS.ComponentMolarFlow(i) * 3600
547             .Cells(topRow + 18, i + 4) = _
548                 PostDistVapOutXHYSYS.ComponentMolarFlow(i) * 3600
549             .Cells(topRow + 19, i + 4) = _
550                 PostReactorOutHYSYS.ComponentMolarFlow(i) * 3600
551             .Cells(topRow + 20, i + 4) = _
552                 PreAdopterHYSYS.ComponentMolarFlow(i) * 3600
553             .Cells(topRow + 21, i + 4) = _
554                 PreCrystallizedRecycleHYSYS.ComponentMolarFlow(i) * 3600
555             .Cells(topRow + 22, i + 4) = _
556                 PreReactorInHYSYS.ComponentMolarFlow(i) * 3600
557             .Cells(topRow + 23, i + 4) = _
558                 ProductHYSYS.ComponentMolarFlow(i) * 3600
559             .Cells(topRow + 24, i + 4) = _
560                 PumpInHYSYS.ComponentMolarFlow(i) * 3600
561             .Cells(topRow + 25, i + 4) = _
562                 ReactorInHYSYS.ComponentMolarFlow(i) * 3600
563             .Cells(topRow + 26, i + 4) = _
564                 ReactorOutHYSYS.ComponentMolarFlow(i) * 3600
565             .Cells(topRow + 27, i + 4) = _
566                 ValveOutHYSYS.ComponentMolarFlow(i) * 3600
567         Next
568
569         .Cells(topRow + 0, 13) = _
570             AdditionalAirHYSYS.molarFlow.GetValue("kgmole/h")
571         .Cells(topRow + 1, 13) = _
572             AdopterHYSYS.molarFlow.GetValue("kgmole/h")
573         .Cells(topRow + 2, 13) = _
574             BoilerBottomOutHYSYS.molarFlow.GetValue("kgmole/h")
575         .Cells(topRow + 3, 13) = _
576             BoilerTopOutHYSYS.molarFlow.GetValue("kgmole/h")
577         .Cells(topRow + 4, 13) = _
578             CrystallizationInHYSYS.molarFlow.GetValue("kgmole/h")
579         .Cells(topRow + 5, 13) = _
580             CrystallizedRecycleHYSYS.molarFlow.GetValue("kgmole/h")
581         .Cells(topRow + 6, 13) = _
582             DecanterOutHYSYS.molarFlow.GetValue("kgmole/h")
583         .Cells(topRow + 7, 13) = _
584             DistillationInHYSYS.molarFlow.GetValue("kgmole/h")
585         .Cells(topRow + 8, 13) = _
586             DistillationLiqOutHYSYS.molarFlow.GetValue("kgmole/h")
587         .Cells(topRow + 9, 13) = _
588             DistillationVapOutHYSYS.molarFlow.GetValue("kgmole/h")
589         .Cells(topRow + 10, 13) = _
590             DistVapOutXHYSYS.molarFlow.GetValue("kgmole/h")
591         .Cells(topRow + 11, 13) = _
```

```
592         ExternalOutHYSYS.molarFlow.GetValue("kgmole/h")
593     .Cells(topRow + 12, 13) = _
594         ExtractedBzAHYSYS.molarFlow.GetValue("kgmole/h")
595     .Cells(topRow + 13, 13) = _
596         ExtractedOthersHYSYS.molarFlow.GetValue("kgmole/h")
597     .Cells(topRow + 14, 13) = _
598         ExtractedWaterHYSYS.molarFlow.GetValue("kgmole/h")
599     .Cells(topRow + 15, 13) = _
600         FeedHYSYS.molarFlow.GetValue("kgmole/h")
601     .Cells(topRow + 16, 13) = _
602         MixerInHYSYS.molarFlow.GetValue("kgmole/h")
603     .Cells(topRow + 17, 13) = _
604         PostDistillationVapOutHYSYS.molarFlow.GetValue("kgmole/h")
605     .Cells(topRow + 18, 13) = _
606         PostDistVapOutXHYSYS.molarFlow.GetValue("kgmole/h")
607     .Cells(topRow + 19, 13) = _
608         PostReactorOutHYSYS.molarFlow.GetValue("kgmole/h")
609     .Cells(topRow + 20, 13) = _
610         PreAdopterHYSYS.molarFlow.GetValue("kgmole/h")
611     .Cells(topRow + 21, 13) = _
612         PreCrystallizedRecycleHYSYS.molarFlow.GetValue("kgmole/h")
613     .Cells(topRow + 22, 13) = _
614         PreReactorInHYSYS.molarFlow.GetValue("kgmole/h")
615     .Cells(topRow + 23, 13) = _
616         ProductHYSYS.molarFlow.GetValue("kgmole/h")
617     .Cells(topRow + 24, 13) = _
618         PumpInHYSYS.molarFlow.GetValue("kgmole/h")
619     .Cells(topRow + 25, 13) = _
620         ReactorInHYSYS.molarFlow.GetValue("kgmole/h")
621     .Cells(topRow + 26, 13) = _
622         ReactorOutHYSYS.molarFlow.GetValue("kgmole/h")
623     .Cells(topRow + 27, 13) = _
624         ValveOutHYSYS.molarFlow.GetValue("kgmole/h")
625
626     .Cells(topRow + 0, 14) = _
627       AdditionalAirHYSYS.StdLiqVolFlow.GetValue("m3/h")
628     .Cells(topRow + 1, 14) = _
629       AdopterHYSYS.StdLiqVolFlow.GetValue("m3/h")
630     .Cells(topRow + 2, 14) = _
631       BoilerBottomOutHYSYS.StdLiqVolFlow.GetValue("m3/h")
632     .Cells(topRow + 3, 14) = _
633       BoilerTopOutHYSYS.StdLiqVolFlow.GetValue("m3/h")
634     .Cells(topRow + 4, 14) = _
635       CrystallizationInHYSYS.StdLiqVolFlow.GetValue("m3/h")
636     .Cells(topRow + 5, 14) = _
637       CrystallizedRecycleHYSYS.StdLiqVolFlow.GetValue("m3/h")
638     .Cells(topRow + 6, 14) = _
639       DecanterOutHYSYS.StdLiqVolFlow.GetValue("m3/h")
```

```
640     .Cells(topRow + 7, 14) = _
641       DistillationInHYSYS.StdLiqVolFlow.GetValue("m3/h")
642     .Cells(topRow + 8, 14) = _
643       DistillationLiqOutHYSYS.StdLiqVolFlow.GetValue("m3/h")
644     .Cells(topRow + 9, 14) = _
645       DistillationVapOutHYSYS.StdLiqVolFlow.GetValue("m3/h")
646     .Cells(topRow + 10, 14) = _
647       DistVapOutXHYSYS.StdLiqVolFlow.GetValue("m3/h")
648     .Cells(topRow + 11, 14) = _
649       ExternalOutHYSYS.StdLiqVolFlow.GetValue("m3/h")
650     .Cells(topRow + 12, 14) = _
651       ExtractedBzAHYSYS.StdLiqVolFlow.GetValue("m3/h")
652     .Cells(topRow + 13, 14) = _
653       ExtractedOthersHYSYS.StdLiqVolFlow.GetValue("m3/h")
654     .Cells(topRow + 14, 14) = _
655       ExtractedWaterHYSYS.StdLiqVolFlow.GetValue("m3/h")
656     .Cells(topRow + 15, 14) = _
657       FeedHYSYS.StdLiqVolFlow.GetValue("m3/h")
658     .Cells(topRow + 16, 14) = _
659       MixerInHYSYS.StdLiqVolFlow.GetValue("m3/h")
660     .Cells(topRow + 17, 14) = _
661       PostDistillationVapOutHYSYS.StdLiqVolFlow.GetValue("m3/h")
662     .Cells(topRow + 18, 14) = _
663       PostDistVapOutXHYSYS.StdLiqVolFlow.GetValue("m3/h")
664     .Cells(topRow + 19, 14) = _
665       PostReactorOutHYSYS.StdLiqVolFlow.GetValue("m3/h")
666     .Cells(topRow + 20, 14) = _
667       PreAdopterHYSYS.StdLiqVolFlow.GetValue("m3/h")
668     .Cells(topRow + 21, 14) = _
669       PreCrystallizedRecycleHYSYS.StdLiqVolFlow.GetValue("m3/h")
670     .Cells(topRow + 22, 14) = _
671       PreReactorInHYSYS.StdLiqVolFlow.GetValue("m3/h")
672     .Cells(topRow + 23, 14) = _
673       ProductHYSYS.StdLiqVolFlow.GetValue("m3/h")
674     .Cells(topRow + 24, 14) = _
675       PumpInHYSYS.StdLiqVolFlow.GetValue("m3/h")
676     .Cells(topRow + 25, 14) = _
677       ReactorInHYSYS.StdLiqVolFlow.GetValue("m3/h")
678     .Cells(topRow + 26, 14) = _
679       ReactorOutHYSYS.StdLiqVolFlow.GetValue("m3/h")
680     .Cells(topRow + 27, 14) = _
681       ValveOutHYSYS.StdLiqVolFlow.GetValue("m3/h")
682
683     .Cells(topRow + 0, 15) = _
684       AdditionalAirHYSYS.HeatFlow.GetValue("GJ/h")
685     .Cells(topRow + 1, 15) = _
686       AdopterHYSYS.HeatFlow.GetValue("GJ/h")
687     .Cells(topRow + 2, 15) = _
```

```
688       BoilerBottomOutHYSYS.HeatFlow.GetValue("GJ/h")
689     .Cells(topRow + 3, 15) = _
690       BoilerTopOutHYSYS.HeatFlow.GetValue("GJ/h")
691     .Cells(topRow + 4, 15) = _
692       CrystallizationInHYSYS.HeatFlow.GetValue("GJ/h")
693     .Cells(topRow + 5, 15) = _
694       CrystallizedRecycleHYSYS.HeatFlow.GetValue("GJ/h")
695     .Cells(topRow + 6, 15) = _
696       DecanterOutHYSYS.HeatFlow.GetValue("GJ/h")
697     .Cells(topRow + 7, 15) = _
698       DistillationInHYSYS.HeatFlow.GetValue("GJ/h")
699     .Cells(topRow + 8, 15) = _
700       DistillationLiqOutHYSYS.HeatFlow.GetValue("GJ/h")
701     .Cells(topRow + 9, 15) = _
702       DistillationVapOutHYSYS.HeatFlow.GetValue("GJ/h")
703     .Cells(topRow + 10, 15) = _
704       DistVapOutXHYSYS.HeatFlow.GetValue("GJ/h")
705     .Cells(topRow + 11, 15) = _
706       ExternalOutHYSYS.HeatFlow.GetValue("GJ/h")
707     .Cells(topRow + 12, 15) = _
708       ExtractedBzAHYSYS.HeatFlow.GetValue("GJ/h")
709     .Cells(topRow + 13, 15) = _
710       ExtractedOthersHYSYS.HeatFlow.GetValue("GJ/h")
711     .Cells(topRow + 14, 15) = _
712       ExtractedWaterHYSYS.HeatFlow.GetValue("GJ/h")
713     .Cells(topRow + 15, 15) = _
714       FeedHYSYS.HeatFlow.GetValue("GJ/h")
715     .Cells(topRow + 16, 15) = _
716       MixerInHYSYS.HeatFlow.GetValue("GJ/h")
717     .Cells(topRow + 17, 15) = _
718       PostDistillationVapOutHYSYS.HeatFlow.GetValue("GJ/h")
719     .Cells(topRow + 18, 15) = _
720       PostDistVapOutXHYSYS.HeatFlow.GetValue("GJ/h")
721     .Cells(topRow + 19, 15) = _
722       PostReactorOutHYSYS.HeatFlow.GetValue("GJ/h")
723     .Cells(topRow + 20, 15) = _
724       PreAdopterHYSYS.HeatFlow.GetValue("GJ/h")
725     .Cells(topRow + 21, 15) = _
726       PreCrystallizedRecycleHYSYS.HeatFlow.GetValue("GJ/h")
727     .Cells(topRow + 22, 15) = _
728       PreReactorInHYSYS.HeatFlow.GetValue("GJ/h")
729     .Cells(topRow + 23, 15) = _
730       ProductHYSYS.HeatFlow.GetValue("GJ/h")
731     .Cells(topRow + 24, 15) = _
732       PumpInHYSYS.HeatFlow.GetValue("GJ/h")
733     .Cells(topRow + 25, 15) = _
734       ReactorInHYSYS.HeatFlow.GetValue("GJ/h")
735     .Cells(topRow + 26, 15) = _
```

```
736       ReactorOutHYSYS.HeatFlow.GetValue("GJ/h")
737     .Cells(topRow + 27, 15) = _
738       ValveOutHYSYS.HeatFlow.GetValue("GJ/h")
739
740
741     .Cells(10, 4) = reactorErr
742     .Cells(11, 4) = crystallizerErr
743
744
745     .Cells(topRow + 47, 6) = CondenserHYSYS.HeatFlow.GetValue("MJ/h")
746     .Cells(topRow + 47, 7) = ToCond.Temperature.GetValue("C")
747     .Cells(topRow + 47, 8) = Reflux.Temperature.GetValue("C")
748
749     .Cells(topRow + 48, 6) = _
750       DistVapOutExpandHYSYS.HeatFlow.GetValue("MJ/h")
751     .Cells(topRow + 48, 7) = _
752       DistillationVapOutHYSYS.Temperature.GetValue("C")
753     .Cells(topRow + 48, 8) = _
754       PostDistillationVapOutHYSYS.Temperature.GetValue("C")
755
756     .Cells(topRow + 49, 6) = PowerOfPumpHYSYS.HeatFlow.GetValue("MJ/h")
757     .Cells(topRow + 49, 7) = PumpInHYSYS.Temperature.GetValue("C")
758     .Cells(topRow + 49, 8) = PreReactorInHYSYS.Temperature.GetValue("C")
759
760     .Cells(topRow + 50, 6) = _
761        PreCrystallizedRecycleHeaterHYSYS.HeatFlow.GetValue("MJ/h")
762     .Cells(topRow + 50, 7) = _
763        PreCrystallizedRecycleHYSYS.Temperature.GetValue("C")
764     .Cells(topRow + 50, 8) = _
765        CrystallizedRecycleHYSYS.Temperature.GetValue("C")
766
767     .Cells(topRow + 51, 6) = PreReactorInHeaterHYSYS.HeatFlow.GetValue("MJ/h")
768     .Cells(topRow + 51, 7) = PreReactorInHYSYS.Temperature.GetValue("C")
769     .Cells(topRow + 51, 8) = ReactorInHYSYS.Temperature.GetValue("C")
770
771     .Cells(topRow + 52, 6) = ValveOutCoolerHYSYS.HeatFlow.GetValue("MJ/h")
772     .Cells(topRow + 52, 7) = DistillationLiqOutHYSYS.Temperature.GetValue("C")
773     .Cells(topRow + 52, 8) = ValveOutHYSYS.Temperature.GetValue("C")
774
775 ' .Cells(topRow + 53, 6) = ReactorOutCoolerHYSYS.HeatFlow.GetValue("MJ/h")
776 ' .Cells(topRow + 53, 7) = ReactorOutHYSYS.Temperature.GetValue("C")
777 ' .Cells(topRow + 53, 8) = DistillationInHYSYS.Temperature.GetValue("C")
778
779     .Cells(topRow + 54, 6) = ReboilerHYSYS.HeatFlow.GetValue("MJ/h")
780     .Cells(topRow + 54, 8) = BoilUp.Temperature.GetValue("C")
781     .Cells(topRow + 54, 7) = ToReb.Temperature.GetValue("C")
782
783     .Cells(topRow + 55, 6) = -((.Cells(topRow + 25, 15).Value + _
```

```
                .Cells(topRow + 28, 15).Value + .Cells(topRow + 30, 15).Value) _
                - (.Cells(topRow + 26).Value + .Cells(topRow + 29, 15).Value)) _
                * 1000
    .Cells(topRow + 55, 7) = reactorTemp - 273.15
    .Cells(topRow + 55, 8) = reactorTemp - 273.15 + 0.1

' .Cells(topRow + 60, 6) = ReactorOutPumpHYSYS.HeatFlow.GetValue("MJ/h")
    .Cells(topRow + 60, 6) = 0
    .Cells(topRow + 60, 7) = PostReactorOutHYSYS.Temperature.GetValue("C")
    .Cells(topRow + 60, 8) = DistillationInHYSYS.Temperature.GetValue("C")

    .Cells(topRow + 61, 6) = DistVapOutCoolerHYSYS.HeatFlow.GetValue("MJ/h")
    .Cells(topRow + 61, 7) = DistVapOutXHYSYS.Temperature.GetValue("C")
    .Cells(topRow + 61, 8) = PostDistVapOutXHYSYS.Temperature.GetValue("C")

    .Cells(topRow + 70, 6) = ExternalOutCoolerHYSYS.HeatFlow.GetValue("MJ/h")

End With
'-------------------------------------------------------------------------------

'-------------------------------------------------------------------------------
Dim CrystallizationInEnthalpy As Double '[MJ/h]
Dim ProductEnthalpy As Double '[MJ/h]
Dim PreCrystallizedRecycleEnthalpy As Double '[MJ/h]

CrystallizationInEnthalpy = CrystallizationInHYSYS. _
                EnthalpyEstimate.GetValue("MJ/kgmole") * _
                CrystallizationInHYSYS.molarFlow.GetValue("kgmole/h") / 1000
ProductEnthalpy = ProductHYSYS.EnthalpyEstimate.GetValue("MJ/kgmole") * _
                ProductHYSYS.molarFlow.GetValue("kgmole/h") / 1000
PreCrystallizedRecycleEnthalpy = PreCrystallizedRecycleHYSYS. _
                EnthalpyEstimate.GetValue("MJ/kgmole") * _
                PreCrystallizedRecycleHYSYS.molarFlow.GetValue("kgmole/h") _
                / 1000


With resultSheet
    .Cells(55, 27) = PowerOfPumpHYSYS.Power.GetValue("kW")
    .Cells(55, 25) = ExternalOutCoolerHYSYS.HeatFlow.GetValue("MJ/h")
    .Cells(55, 35) = DistVapOutExpandHYSYS.Power.GetValue("kW")

    .Cells(topRow + 59, 6).Value = -(ProductEnthalpy + _
                        PreCrystallizedRecycleEnthalpy - _
                        CrystallizationInEnthalpy) * 1000
    .Cells(topRow + 59, 8).Value = crystallizerTemp
End With


```

```
832 Call RunPython("import heatexchange; heatexchange.main()") '
833 ThisWorkbook.Save
834 '-------------------------------------------------------------------------------
835
836 Debug.Print "completed"
837 End Sub
```

ソースコード B.6　抽出塔

```
1  Option Explicit
2
3  Sub main(outletFlow As Variant, _
4          BzAFlow As Variant, _
5          OthersFlow As Variant, _
6          WaterFlow As Variant, _
7          T As Double)
8
9  Dim toluene As Double '[kmol/h]
10 Dim balc As Double '[kmol/h]
11 Dim bald As Double '[kmol/h]
12 Dim bza As Double '[kmol/h]
13 Dim h2o As Double '[kmol/h]
14 Dim Nitrogen As Double '[kmol/h]
15 Dim Oxygen As Double '[kmol/h]
16 Dim co As Double '[kmol/h]
17 Dim co2 As Double '[kmol/h]
18 Dim HotWater As Double '[kmol/h]
19
20 Dim Csat As Double '水単位質量当たりの安息香酸の溶解度 [BzA-g/ H2O-g]
21
22
23 ' 蒸留塔出口の成分別モル数を求める.
24 toluene = outletFlow(0)
25 balc = outletFlow(1)
26 bald = outletFlow(2)
27 bza = outletFlow(3)
28 h2o = outletFlow(4)
29 Nitrogen = outletFlow(5)
30 Oxygen = outletFlow(6)
31 co = outletFlow(7)
32 co2 = outletFlow(8)
33
34 ' 安息香酸のモル量すべてが溶ける水の量を溶解度と温度から求める.
35 Csat = _
36   (0.0000203 * T ^ 4 + 0.000297 * T ^ 3 + 0.047 * T ^ 2 + 1.43 * T + 24.71) / _
37   1000
38 HotWater = ((bza * BzAMolarMass * 1000) / Csat) / WaterMolarMass / 1000
39
40
```

```
41 ' 抽出器で，100%分離できたと考え,成分別モル流量を求める.
42 BzAFlow(3) = bza
43
44 OthersFlow(0) = toluene
45 OthersFlow(1) = balc
46 OthersFlow(2) = bald
47 OthersFlow(4) = h2o
48 OthersFlow(5) = Nitrogen
49 OthersFlow(6) = Oxygen
50 OthersFlow(7) = co
51 OthersFlow(8) = co2
52
53 WaterFlow(4) = HotWater
54
55 End Sub
```

ソースコード B.7　晶析器

```
1  Option Explicit
2
3  '---------------変数定義-----------------------------------------------------
4
5  Private Const j_pow = 1.78
6  Private Const b_pow = 1.2
7  Private Const g_pow = 0.44
8  Private Const R = 8.314
9  Private Const kv = 0.1 '結晶体積形状係数
10
11 Private kb As Double '核発生速度
12 Private kg As Double '結晶成長速度
13 Private kg0 As Double '結晶成長頻度因子
14 Private Eg As Double '結晶成長活性化エネルギー
15 Private rhoc As Double '結晶密度
16 Private GrowthRate As Double '結晶成長速度定数
17 Private T_MSMPR As Double '連続晶析器内温度
18 Private T_MSMPR_K As Double '連続晶析器内温度
19 Private C_Feed As Double 'Feed 液 Bac 濃度
20 Private Csat As Double 'Bac 飽和濃度
21 Private delta_C As Double '過飽和度
22 Private Mt As Double '懸濁密度（体積当たり結晶生産速度）
23 Private B0 As Double '一次核発生速度
24 Private rhoL As Double '液溶媒密度
25 Private tau_MSMPR As Double '平均滞留時間
26 Private n0 As Double '個数密度
27 Private Vol_MSMPR As Double '晶析装置液体積
28 Private v0_MSMPR As Double 'Feed 液体積流量
29 Private Yc As Double '全体結晶転化率
30 Private Ycmax As Double '可能結晶転化率
31
```

```
Function MSMPR(T_input As Double, _
                C_Feed_input As Double, _
                v0_input As Double, _
                Vol_input As Double, _
                BzAMolarFlow As Double) As Double

'諸条件から結晶生産量を求めます.

'--------------操作条件入力-------------------------------------------------
T_MSMPR = T_input '[℃]
C_Feed = C_Feed_input '[g/g]
v0_MSMPR = v0_input '[m3/h]
Vol_MSMPR = Vol_input '[m3]

'--------------パラメータ入力-----------------------------------------------
Eg = 40.05 '[kJ/mol]
kg0 = 10600000# '[μm/s]
kb = 9160000000000# '[(#/m3 s)(g/mL)^j(BzA-g/H2O-g)^b]
rhoc = 1.32 '[g/cm3]
rhoL = 0.998 '[g/mL]

'--------------諸量計算-----------------------------------------------------
tau_MSMPR = Vol_MSMPR / v0_MSMPR * 60 '[min]
Csat = (0.0000203 * T_MSMPR ^ 4 + 0.000297 * T_MSMPR ^ 3 + _
    0.047 * T_MSMPR ^ 2 + 1.43 * T_MSMPR + 24.71) / 1000 '[g/g]
T_MSMPR_K = T_MSMPR + 273.15 '[K]
kg = kg0 * Exp(-(Eg * 1000) / (R * T_MSMPR_K)) '[μm/s]


'----------晶析器内濃度C の決定(濃度誤差の二分法)-------------------------------

Dim C1 As Double, C2 As Double, C3 As Double '仮定晶析器内濃度
Dim Cnew1 As Double, Cnew2 As Double, Cnew3 As Double '更新濃度
Dim err1 As Double, err2 As Double, err3 As Double '誤差
Dim counter As Integer '計算回数
Dim eps As Double '収束半径

eps = 0.0000001
counter = 0
C1 = Csat * 1.00000001 '下限
C2 = C_Feed * 0.99999999 '上限

Call calc(C1, Cnew1, err1) '小さい濃度C1 での誤差検出
Call calc(C2, Cnew2, err2) '大きい濃度C2 での誤差検出

Do While (Abs((C1 - C2) / C1) > eps)

    C3 = (C1 + C2) / 2
```

```
    Call calc(C3, Cnew3, err3) '中間濃度C3 での誤差検出
    '更新
    If err1 * err3 >= 0 Then
        C1 = C3
        err1 = err3
    Else
        C2 = C3
        err2 = err3
    End If
    counter = counter + 1
Loop

'---------------出力部-----------------------------------------------------------
Yc = (C_Feed - C3) / C_Feed * 100 '[wt%]
Ycmax = (C_Feed - C3) / (C_Feed - Csat) * 100 '[wt%]

MSMPR = BzAMolarFlow * Yc * Ycmax / 10000
'MSMPR = (C_Feed - C3) * rhoL * v0_MSMPR * 1000 '[kg/h]
End Function

Sub calc(ByRef C As Double, ByRef Cnew As Double, ByRef err As Double)
'計算ユニットです

Dim Mtnew As Double

Mt = (C_Feed - C) * rhoL '定義式 [g/mL]
delta_C = C - Csat '定義式 [g/g]
GrowthRate = kg * delta_C ^ g_pow '実験式 [μm/s]
B0 = kb * Mt ^ j_pow * delta_C ^ b_pow '実験式 [#/m3 s]
n0 = B0 / GrowthRate '定義式 [m3 μm]
Mtnew = 6 * kv * rhoc * (n0 * 1000000) * _
        ((GrowthRate * 0.000001) * (tau_MSMPR * 60)) ^ 4 '個数収支式 [g/mL]
Cnew = C_Feed - Mtnew / rhoL '個数収支式 [g/g]
err = C - Cnew '誤差

End Sub
```

ソースコード B.8　関数群

```
Option Explicit

Sub solve_k(Temp As Double, k As Variant)

Dim i As Integer

Dim a As Variant
Dim E As Variant
Dim R As Double

```

```
11 a = Array(Exp(20.634), Exp(17.928), Exp(15.4), Exp(19.698))
12 E = Array(81.389, 69.53, 56.987, 71.442)
13 k = Array(0, 0, 0, 0)
14 R = 8.314
15
16 For i = 0 To 3
17     k(i) = a(i) * Exp(-E(i) * 1000 / R / Temp)
18 Next i
19
20 End Sub
21
22 Sub solveX(ByRef F As Variant, ByRef x As Variant)
23
24 Dim i As Integer
25 Dim sum As Long
26
27 sum = sumOfFlow(F)
28
29 For i = 0 To 8
30     x(i) = F(i) / sum
31 Next i
32
33 End Sub
34
35 Function sumOfFlow(F As Variant) As Double
36
37 Dim i As Integer
38
39 sumOfFlow = 0
40
41 For i = 0 To 8
42     sumOfFlow = sumOfFlow + F(i)
43 Next i
44
45 End Function
46
47 Sub solveComponentVolumeFlow(molarFlow As Variant, volumeFlow As Variant)
48
49 Dim i As Integer
50 Dim param As Variant
51
52 ' [m3/mol]
53 param = Array(0.10550147646805, 0.10002241059429, 0.101176788713442, _
54     0.112399721649694, 1.77522405788194E-02, 0, 0, 0, 5.32284437764362E-02)
55
56 For i = 0 To 8
57     volumeFlow(i) = molarFlow(i) * param(i)
58 Next
```

```
59
60 End Sub
```

ソースコード B.9　最適化を行うコード

```
1  Option Explicit
2
3  Sub gridrun()
4
5  Dim topRow1 As Integer
6  Dim topRow2 As Integer
7
8  Dim leftColumn1 As Integer
9  Dim leftColumn2 As Integer
10
11 Dim resultSheet As Worksheet
12 Dim writeSheet As Worksheet
13 Dim saveSheet As Worksheet
14 Set resultSheet = ThisWorkbook.Sheets(1)
15 Set writeSheet = ThisWorkbook.Sheets(4)
16 Set saveSheet = ThisWorkbook.Sheets(5)
17
18 Dim crVolume As Double
19 Dim reVolume As Double
20
21
22
23 Dim i As Integer
24 Dim j As Integer
25 Dim p As Integer
26 Dim l As Integer
27
28 topRow1 = 8
29 leftColumn1 = 4
30
31 topRow2 = 3
32 leftColumn2 = 2
33
34 For i = 1 To 1
35     j = 0
36
37     Call process_all.main
38
39     With writeSheet
40         .Cells(topRow1 + j, leftColumn1 + i) = resultSheet.Cells(28, 22)
41
42         If (resultSheet.Cells(10, 4) > 0.001) Then
43             If (resultSheet.Cells(11, 4) > 0.001) Then
44                 .Cells(topRow1 + j, leftColumn1 + i).Interior.Color = _
```

```
                        RGB(255, 0, 0)
            Else
                .Cells(topRow1 + j, leftColumn1 + i).Interior.Color = _
                        RGB(0, 255, 0)
            End If
        Else
            If (resultSheet.Cells(11, 4) > 0.001) Then
                .Cells(topRow1 + j, leftColumn1 + i).Interior.Color = _
                        RGB(0, 0, 255)
            End If
        End If
    End With
Next i

End Sub
```