

触媒懇談会ニュース

触媒学会シニア懇談会

エンジニアはAI とどう向き合うか

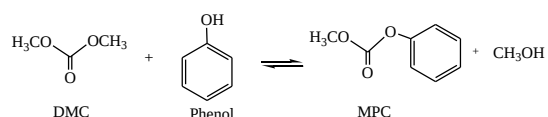
常木英昭

この 10 年ほど化学工学会関西支部の実践化学工学セミナーの反応工学(応用編)の講師をしている。応用編ということで実際のデータの解析や反応器設計のために具体的な計算ができるように Excel でのソルバーなどを使った解析のノウハウなどについて解説をしていた。反応器の設計については微分方程式を解く必要があるのでプログラムを組むのが良いのだが、どの言語を選ぶか、学習コストが高いなどの問題から積極的には取り扱ってこなかった。そうしたなか 5 年ほど前から学習コストが低く、科学技術計算ライブラリの充実した Python をよく使うようになったが、学習しやすいとは言ってもそれなり大変なこと、プログラミング自体に抵抗感のある方も多いため、これまではコードと計算結果の提示にとどめていた。しかしこの 1 年ほどの生成 AI の進歩が著しく、考え方を指示するだけで質の良いコードを出力するようになり、自分で作ったコードと遜色なくなっている。このような AI に対してエンジニアはどう向き合っていけば良いのか、皆さんはどうお考えになるでしょうか。

具体的事例の紹介

私のテキストから具体的な事例を紹介する。解くべき問題は、平衡を考慮する必要のある

る液相均一触媒を用いた 2 次反応(エステル交換反応)であり、次のような反応である



メチルフェニルカーボネート(MPC)合成である。ジメチルカーボネート(DMC)とフェノール(PhOH)のエステル交換反応で MPC とメタノール(MeOH)が生成する。

テキストでは微分法と積分法(解析解を用いる)に分けて、Excel を使って解析手法を解説している。

これを ChatGPT Plus (一応課金) に対して、順番に解析手法を指示しながら解析を進めていく。まず解析解を用いた積分法について最初のプロンプトは

(以下プロンプトは罫線で囲み青文字表記)

平衡を考慮する必要のある液相均一触媒を用いた 2 次反応(エステル交換反応)の反応速度解析について：

反応はポリカーボネート製造の中間体であるメチルフェニルカーボネート(MPC)合成である。ジメチルカーボネート(DMC)とフェノール(PhOH)のエステル交換反応で MPC とメタノール(MeOH)が生成する。反応条件：触媒 $\text{Ti}(\text{OPh})_4$ 0.1 mmol L^{-1} 、反応温度 200°C 、DMC/PhOH モル濃度比

=5.8106/5.8081.

反応結果は 反応時間 [h] 0.5, 1,

1.5, 2, 2.5, 3, 3.5, 4, 24

MPC 収率 [-] 0.011, 0.0215, 0.0304,

0.0361, 0.0403, 0.0442, 0.0449, 0.0461,

0.0474

正反応の速度定数を k_1 , 逆反応の速度定数を k_2 とし, DMC, PhOH の初期濃度を C_{A0} , C_{B0} , 時間 t における MPC 収率を Y とすると, DMC の濃度 $C_A = C_{A0}(1-Y)$ であり, 時間 t における各成分の濃度は

DMC, PhOH, MPC, MeOH:

$C_A, C_{B0}-C_{A0}+C_A, C_{A0}-C_A, C_{A0}-C_A$ となる.

反応速度が各成分の濃度に比例する 2 次反応とした場合の反応速度式は

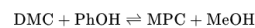
$-dC_A/dt = k_1 \cdot C_A \cdot (C_{B0} - C_{A0} + C_A) - k_2 \cdot (C_{A0} - C_A)^2$

となる。

平衡濃度を C_{Ae} とした場合、 k_2 を k_1 と C_{A0}, C_{B0}, C_{Ae} で表してください。

これに対して、指示を整理しながら回答してくれる。少し冗長ですがすべて示す。

与えられた可逆反応：



に対し、反応速度式が

$$-\frac{dC_A}{dt} = k_1 C_A (C_{B0} - C_{A0} + C_A) - k_2 (C_{A0} - C_A)^2$$

と与えられている。

【目標】

平衡時の DMC 濃度を C_{Ae} とすると、

$$\left. \frac{dC_A}{dt} \right|_{C_A=C_{Ae}} = 0$$

を用いて、逆反応の速度定数 k_2 を $k_1, C_{A0}, C_{B0}, C_{Ae}$ で表す。

【導出】

平衡時に速度がゼロであることから、

$$k_1 C_{Ae} (C_{B0} - C_{A0} + C_{Ae}) = k_2 (C_{A0} - C_{Ae})^2$$

この式を変形して k_2 を求める：

$$k_2 = \frac{k_1 C_{Ae} (C_{B0} - C_{A0} + C_{Ae})}{(C_{A0} - C_{Ae})^2}$$

【結論】

$$k_2 = \frac{k_1 C_{Ae} (C_{B0} - C_{A0} + C_{Ae})}{(C_{A0} - C_{Ae})^2}$$

この式により、正反応の速度定数 k_1 、初期濃度 C_{A0}, C_{B0} 、および平衡時の DMC 濃度 C_{Ae} が分かれば、逆反応の速度定数 k_2 を一意に決定できる。

のようにまず式の変形をしてくれた。

反応速度式の k_2 を求めた式で置換してください。

了解しました。

以下に、反応速度式：

$$-\frac{dC_A}{dt} = k_1 C_A (C_{B0} - C_{A0} + C_A) - k_2 (C_{A0} - C_A)^2$$

に、先ほど導出した

$$k_2 = \frac{k_1 C_{Ae} (C_{B0} - C_{A0} + C_{Ae})}{(C_{A0} - C_{Ae})^2}$$

を代入した式を示します。

【置換後の反応速度式】

$$-\frac{dC_A}{dt} = k_1 C_A (C_{B0} - C_{A0} + C_A) - \left(\frac{k_1 C_{Ae} (C_{B0} - C_{A0} + C_{Ae})}{(C_{A0} - C_{Ae})^2} \right) (C_{A0} - C_A)^2$$

【整理後の形（任意で係数くくり出し）】

$$-\frac{dC_A}{dt} = k_1 \left[C_A (C_{B0} - C_{A0} + C_A) - \frac{C_{Ae} (C_{B0} - C_{A0} + C_{Ae})}{(C_{A0} - C_{Ae})^2} (C_{A0} - C_A)^2 \right]$$

これにより、未知数を k_1 のみに整理した反応速度式となり、回帰分析などに用いやすくなります。必要に応じて、MPC 収率 Y による変数変換も可能です。ご希望があればそれに対応します。

この微分方程式は変数分離型なので解析的に解けます。時間 t の時の C_A を求めてください。

ご指摘の通り、この反応速度式は変数分離型であり、解析的に積分できます。以下に詳細な導出手順を示します。

与えられた反応速度式 (k_2 を置換済)

$$-\frac{dC_A}{dt} = k_1 \left[C_A (C_{B0} - C_{A0} + C_A) - \frac{C_{Ae} (C_{B0} - C_{A0} + C_{Ae})}{(C_{A0} - C_{Ae})^2} (C_{A0} - C_A)^2 \right]$$

以下では、右辺を簡略化するために記号を定義します。

記号の定義

- $C_A(t)$: 時間 t における DMC 濃度
- C_{A0} : DMC 初期濃度
- C_{B0} : PhOH 初期濃度
- C_{Ae} : DMC の平衡濃度
- k_1 : 正反応速度定数
- 記号簡略化のため：

$$\alpha = C_{B0} - C_{A0}, \quad \beta = \frac{C_{Ae} (C_{B0} - C_{A0} + C_{Ae})}{(C_{A0} - C_{Ae})^2}$$

変数分離と積分

反応速度式を変数分離型で書き直します：

$$\frac{dC_A}{C_A(\alpha + C_A) - \beta(C_{A0} - C_A)^2} = -k_1 dt$$

この分母は 2 次関数と 2 次関数の差ですので、整理が必要です。

分母の展開と整理

- $C_A(\alpha + C_A) = \alpha C_A + C_A^2$
- $(C_{A0} - C_A)^2 = C_{A0}^2 - 2C_{A0}C_A + C_A^2$
- よって：

$$\begin{aligned}\text{分母} &= \alpha C_A + C_A^2 - \beta(C_{A0}^2 - 2C_{A0}C_A + C_A^2) \\ &= (1 - \beta)C_A^2 + (\alpha + 2\beta C_{A0})C_A - \beta C_{A0}^2\end{aligned}$$

これを一般の 2 次式の形で表すと：

$$\frac{dC_A}{aC_A^2 + bC_A + c} = -k_1 dt$$

2 次式の積分公式

積分：

$$\int \frac{dC_A}{aC_A^2 + bC_A + c}$$

は判別式 $D = b^2 - 4ac$ の符号によって次のいずれかになります：

- $D > 0$ ：逆正接関数 ($\tanh^{-1}, \ln$ など)
- $D = 0$ ：単純な分数
- $D < 0$ ：逆三角関数 ($\tan^{-1}$ など)

最終形

よって、解析解の一般形は：

$$t = -\frac{1}{k_1} \int \frac{dC_A}{aC_A^2 + bC_A + c} \Rightarrow C_A(t) = \text{関数の逆関数による陽解 (場合により } \arctan, \ln \text{)}$$

注意点と次のステップ

- 実際に陽解 ($C_A(t) = \dots$) を得るには、上記の係数 a, b, c を具体的に代入し、積分を評価する必要があります。
- 濃度初期値 $C_A(0) = C_{A0}$ を使って積分定数を求めることができます。
- ご希望であれば **具体的な数値** (例：CA0, CB0, CAe) を代入して、log または arctangent を使った積分評価・明示的な解をお示しできます。

ご希望の次のステップをお知らせください：

- ☐ 一般形で陽解の完全導出 (log または $\tan^{-1}$)
- ☐ 数値代入して積分評価 (Python/数式)
- ☐ MPC 収率 Y に変換した式に直す

どれに進めましょうか？

一般形で陽解の完全導出してください

では、与えられた変数分離型の常微分方程式 (ODE) を一般形で解析的に解き、陽解 $C_A(t)$ を導出します。

1. 微分方程式の形 (前回までの結果)

$$\frac{dC_A}{dt} = -k_1 [C_A(C_{B0} - C_{A0} + C_A) - \beta(C_{A0} - C_A)^2] \quad (\text{※ここで } \beta = \frac{C_{A0}(C_{B0} - C_{A0} + C_{Ae})}{(C_{A0} - C_{Ae})^2})$$

右辺を展開すると、

$$\frac{dC_A}{dt} = -k_1 [(1 - \beta)C_A^2 + (\alpha + 2\beta C_{A0})C_A - \beta C_{A0}^2] \quad (\alpha = C_{B0} - C_{A0}) \quad (\star)$$

これを変数分離して、

$$\int \frac{dC_A}{(1 - \beta)C_A^2 + (\alpha + 2\beta C_{A0})C_A - \beta C_{A0}^2} = -k_1 t + C \quad (1)$$

2. 分母の 2 次式の判別式

2 次式：

$$aC_A^2 + bC_A + c, \quad \text{where } \begin{cases} a = 1 - \beta \\ b = \alpha + 2\beta C_{A0} \\ c = -\beta C_{A0}^2 \end{cases}$$

判別式：

$$D = b^2 - 4ac = (\alpha + 2\beta C_{A0})^2 + 4\beta(1 - \beta)C_{A0}^2$$

これは現実的な系で常に正 $\rightarrow$ **実数 2 解あり (分母は因数分解可能)**。

3. 因数分解と置換

$$aC_A^2 + bC_A + c = a(C_A - r_1)(C_A - r_2)$$

ここで、

$$r_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$$

とすれば、

$$\int \frac{dC_A}{a(C_A - r_1)(C_A - r_2)} = -k_1 t + C$$

4. 部分分数分解して積分

$$\frac{1}{a(C_A - r_1)(C_A - r_2)} = \frac{A}{C_A - r_1} + \frac{B}{C_A - r_2}$$

標準形の積分結果は：

$$\int \left(\frac{A}{C_A - r_1} + \frac{B}{C_A - r_2} \right) dC_A = A \ln |C_A - r_1| + B \ln |C_A - r_2|$$

よって：

$$\frac{1}{a} \ln \left| \frac{C_A - r_1}{C_A - r_2} \right| = -k_1(r_1 - r_2)t + C'$$

整理すると：

$$\ln \left| \frac{C_A - r_1}{C_A - r_2} \right| = -ak_1(r_1 - r_2)t + C''$$

5. 初期条件を使って定数決定

初期条件：

$$t = 0 \Rightarrow C_A = C_{A0}$$

$$\ln \left| \frac{C_{A0} - r_1}{C_{A0} - r_2} \right| = C''$$

よって最終形は：

$$\ln \left(\frac{C_A - r_1}{C_A - r_2} \cdot \frac{C_{A0} - r_2}{C_{A0} - r_1} \right) = -ak_1(r_1 - r_2)t$$

6. 最終的な陽解の形 ($C_A(t)$ の式)

両辺を指数関数にして：

$$\frac{C_A - r_1}{C_A - r_2} = \frac{C_{A0} - r_1}{C_{A0} - r_2} \cdot e^{-ak_1(r_1 - r_2)t}$$

これを $C_A(t)$ について解くと：

$$C_A(t) = \frac{r_1 - \left(\frac{C_{A0} - r_1}{C_{A0} - r_2} \right) r_2 e^{-ak_1(r_1 - r_2)t}}{1 - \left(\frac{C_{A0} - r_1}{C_{A0} - r_2} \right) e^{-ak_1(r_1 - r_2)t}}$$

最終結果：一般形の陽解

$$C_A(t) = \frac{r_1 - \left(\frac{C_{A0} - r_1}{C_{A0} - r_2} \right) r_2 e^{-ak_1(r_1 - r_2)t}}{1 - \left(\frac{C_{A0} - r_1}{C_{A0} - r_2} \right) e^{-ak_1(r_1 - r_2)t}}$$

ここで：

- $a = 1 - \beta$
- $b = \alpha + 2\beta C_{A0}$
- $c = -\beta C_{A0}^2$
- $r_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

この式に数値を代入することで、解析的に $C_A(t)$ を算出できます。ご希望あれば Python 実装も可能です。

実験データは t と Y の値の組になっています。 $C_A = C_{A0}(1-Y)$ ですから時間 t での Y を求めることができます。

C_A は上式で計算することができますから k_1 と C_{Ae} を未知数として最小2乗法で k_1 と C_{Ae} を求めることができます。Python でのコードを作成し、 k_1 と C_{Ae} を求め、 Y のデータと計算値のパリティプロットと時間 t を横軸に、縦軸に Y のデータと計算値のプロットを示してください。なお計算値は滑らかな曲線で表示してください。

計算結果に問題がなければ Python のコードをダウンロードできるようにしてください。

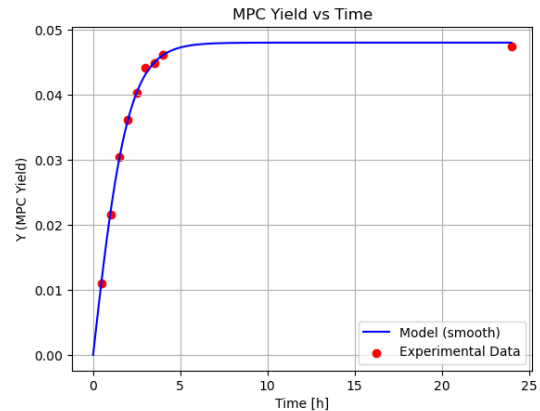
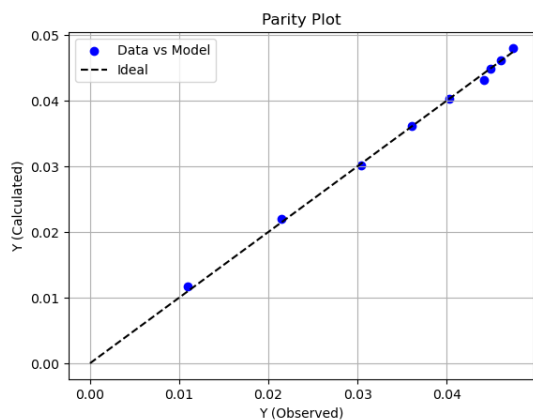
計算結果と Python のコードが表示される。

```
=== 推定パラメータ ===
k1 = 1.157967e-06 L/mol/s
k2 = 4.550436e-04 L/mol/s
CAe = 5.531618 mol/L

=== 時間ごとの Y (実験値 vs 計算値) ===
t = 0.5 h : Y_obs = 0.01100, Y_calc = 0.01174
t = 1.0 h : Y_obs = 0.02150, Y_calc = 0.02201
t = 1.5 h : Y_obs = 0.03040, Y_calc = 0.03018
t = 2.0 h : Y_obs = 0.03610, Y_calc = 0.03618
t = 2.5 h : Y_obs = 0.04030, Y_calc = 0.04035
t = 3.0 h : Y_obs = 0.04420, Y_calc = 0.04313
t = 3.5 h : Y_obs = 0.04490, Y_calc = 0.04494
t = 4.0 h : Y_obs = 0.04610, Y_calc = 0.04609
t = 24.0 h : Y_obs = 0.04740, Y_calc = 0.04801
```

再現性・収束性ともに良好な結果となっています。他にフィッティング関数の切り替えや信頼区間の出力などご希望があれば対応可能です。

得られたコードを実行した結果グラフを示す。



これも冗長になるが出力されたコードを示す。

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

# 実験データ
t_data = np.array([0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 24]) * 3600 # 時間 [s]
Y_data = np.array([0.011, 0.0215, 0.0304, 0.0361, 0.0403, 0.0442, 0.0449, 0.0461, 0.0474])

# 初期濃度
CA0 = 5.8106 # mol/L
CB0 = 5.8081 # mol/L
alpha = CB0 - CA0

def model_CA(t, k1, CAe):
    beta = CAe * (CB0 - CA0 + CAe) / (CA0 - CAe)**2
    a = 1 - beta
    b = alpha + 2 * beta * CA0
    c = -beta * CA0**2
    D = b**2 - 4 * a * c

    if D <= 0 or a == 0:
        return np.full_like(t, CA0) # エラ回避用

    sqrt_D = np.sqrt(D)
    r1 = (-b + sqrt_D) / (2 * a)
    r2 = (-b - sqrt_D) / (2 * a)
    coeff = (CA0 - r1) / (CA0 - r2)
    exp_term = np.exp(-a * k1 * (r1 - r2) * t)
    CA_t = (r1 - coeff * r2 * exp_term) / (1 - coeff * exp_term)
    return CA_t

def objective(params):
    k1, CAe = params
    CA_model = model_CA(t_data, k1, CAe)
    Y_model = 1 - CA_model / CA0
    return np.sum((Y_model - Y_data)**2)

# 初期推定値と境界
initial_guess = [1.2e-6, 5.5316]
bounds = [(1e-8, 1e-4), (1.0, CA0)]

res = minimize(objective, initial_guess, bounds=bounds)

# 推定値
k1_opt, CAe_opt = res.x

# k2 の計算
k2_opt = k1_opt * CAe_opt * (CB0 - CA0 + CAe_opt) / (CA0 - CAe_opt)**2

# 推定値
k1_opt, CAe_opt = res.x

# k2 の計算
k2_opt = k1_opt * CAe_opt * (CB0 - CA0 + CAe_opt) / (CA0 - CAe_opt)**2

# 曲線描画用データ
t_fit = np.linspace(0, 24*3600, 500)
CA_fit = model_CA(t_fit, k1_opt, CAe_opt)
Y_fit = 1 - CA_fit / CA0

# 実験点におけるモデル値
Y_model_points = 1 - model_CA(t_data, k1_opt, CAe_opt) / CA0

# /リテンプロット
plt.figure()
plt.scatter(Y_data, Y_model_points, label='Data vs Model', color='blue')

plt.plot([0, max(Y_data)], [0, max(Y_data)], 'k--', label='Ideal')
plt.xlabel('Y (Observed)')
plt.ylabel('Y (Calculated)')
plt.title('Parity Plot')
plt.legend()
plt.grid(True)

# 時間変化プロット
plt.figure()
plt.plot(t_fit/3600, Y_fit, label='Model (smooth)', color='blue')
plt.scatter(t_data/3600, Y_data, color='red', label='Experimental Data')
plt.xlabel('Time [h]')
plt.ylabel('Y (MPC Yield)')
plt.title('MPC Yield vs Time')
plt.legend()
plt.grid(True)

plt.show()
```

```
# コンソール出力
print("=== 推定パラメータ ===")
print(f"k1 = {k1_opt:.6e} L/mol/s")
print(f"k2 = {k2_opt:.6e} L/mol/s")
print(f"CAe = {CAe_opt:.6f} mol/L\n")

print("=== 時間ごとの Y(実験値 vs 計算値) ===")
for t_h, y_obs, y_calc in zip(t_data / 3600, Y_data, Y_model_points):
    print(f"t = {t_h:>5.1f} h : Y_obs = {y_obs:.5f}, Y_calc = {y_calc:.5f}")
```

式の変形から積分の解析的な実行まで冗長ではあるが思考過程を、段階を追ってレポート形式で分かりやすく示している。私のテキストでは紙面の都合から式の変形とか積分については省略している。

出力されたコードは一箇所使用した関数が最適なものではないが、結果は十分な精度であった。

このように反応工学の基礎知識があれば AI に適切な指示を出すことで Python のプログラミング知識がなくとも（実行環境を整備する必要はあるが）解析コードと解析結果を得ることができる。今回は Python のコードに関する指示は出しておらず、出力形式を指示しただけである。

（ChatGPT の回答と Python のコードは PDF ファイルから画像として貼り付けたため、小さくて見にくくなっている。必要に応じて拡大表示してほしい）

ここまで式の変形や積分の解析解の導出など細かく指示した。これを丸投げしたらどうなるかについて検討した。

ChatGPT ではすでにこの解析を行った後なので、別の AI（Gemini 2.5Pro）を使った。

プロンプト中の反応の説明とデータについては省略する。

与えられたデータから k_1, k_2 を算出してください。
後で検証できるように、Python でコーディングし、ダウンロードできるようにしてください。

出力するのは k_1, k_2 、実験データと対比できるように時間、データ、計算値の表
データと計算値のパリティプロット、横軸 時間 縦軸 収率（データ：点プロット、計算値：滑らかな曲線）のグラフです。
必要な情報が不足しているときは先に質問してください。

回答は画面がダークモードだったのでドキュメントに出力したものを貼り付けた。区別がつくように、色は緑にした。

与えられた反応データに基づき、2 次反応速度式モデルを用いて反応速度定数 k_1 （正反応）と k_2 （逆反応）を算出しました。
最適化計算により得られた反応速度定数は以下の通りです。

- 正反応速度定数 (k_1):
 $0.00493 \text{ L} \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$
- 逆反応速度定数 (k_2):
 $2.09 \text{ L} \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$

解析手法

本解析では、以下の手順で反応速度定数を算出しました。

1. 反応速度式の定義:

反応は、ジメチルカーボネート(DMC)とフェノール(PhOH)からメチルフェニルカーボネート(MPC)とメタノール(MeOH)が生成する可逆な 2 次反応です。

$\text{DMC(A)} + \text{PhOH(B)} \rightleftharpoons \text{MPC(C)} + \text{MeOH(D)}$

反応速度式は、各成分の濃度を用いて以下のように表されます。ここで、 x は生成した MPC の濃度 ($\text{mmol} \cdot \text{L}^{-1}$) です。

$\text{d}t\text{d}x = k_1[\text{DMC}][\text{PhOH}] - k_2[\text{MPC}][\text{MeOH}]$

各成分の濃度は、初期濃度 (C_{A0}, C_{B0}) と反応進行度 x を用いて次のように表現で

きます。

- $[DMC]=C_{A0}-x$
- $[PhOH]=C_{B0}-x$
- $[MPC]=x$
- $[MeOH]=x$

これを代入すると、反応速度式は以下のようになります。

$$\frac{dx}{dt}=k_1(C_{A0}-x)(C_{B0}-x)-k_2 \cdot x^2$$

2. 収率(Y)を用いた表現:

実験データは MPC 収率(Y)で与えられています。収率 Y は、律速反応物であるフェノール(PhOH)の初期濃度 C_{B0} に対する生成した MPC の濃度 x の比で定義されます。 $Y=C_{B0} \cdot x \rightarrow x=Y \cdot C_{B0}$

これを時間で微分すると、 $\frac{dx}{dt} = C_{B0} \cdot \frac{dY}{dt}$ となります。これらの関係式を上記の反応速度式に代入し、収率 Y に関する微分方程式を導出します。

$$\frac{dY}{dt}=k_1(C_{A0}-Y \cdot C_{B0})(1-Y)-k_2 Y \cdot 2C_{B0}$$

3. 数値計算による最適化:

この微分方程式を数値的に解き (SciPy の `solve_ivp` を使用)、計算から得られる収率と実験データの収率との間の誤差 (残差平方和) が最小になるような k_1 と k_2 の組み合わせを、非線形最適化アルゴリズム (SciPy の `minimize`) を用いて探索しました。

解析結果 (うまくいっていないので以下省略)

うまくいかない原因はアルゴリズムにあるのではなく、最適化に使った関数がこの問題に合っていないことによるものと思われる

た。この部分は Python のライブラリ関数に関する知識が必要である。

収率はフェノールに対してではなく DMC に対して定義されています。修正してください

結果が妥当か分かるようにプロットは表示してください。

コードを実行した結果はあまりよくありません。計算方法としては `minimize` 関数を使うのではなく計算値とデータの残差を取り、`least_squares` を使って最適化する方がよいと思います。

ご指摘いただき、誠にありがとうございます。前回の回答に単位表記の誤り、収率の定義の誤認、および数式表現の問題がございました。大変失礼いたしました。

ご指摘に基づき、以下の点を修正して再解析を行いました。

1. **収率(Y)の定義:** DMC の初期濃度 C_{A0} に対する収率に修正しました。
2. **最適化手法:** `minimize` 関数から、カーブフィッティングにより適した `least_squares` 関数に変更しました。
3. **数式・単位表記:** LaTeX のレンダリングを修正し、化学反応式と単位を正しく表示するようにしました。
4. **プロット表示:** コード実行時にグラフが直接表示されるようにし、結果の妥当性を視覚的に判断しやすくしました。

修正・再解析の結果

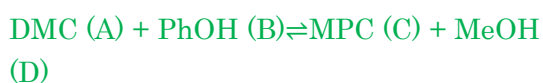
修正したモデルと最適化手法を用いて算出した反応速度定数は以下の通りです。

- 正反応速度定数 (k_1):
 $0.00381 \text{ L} \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$
- 逆反応速度定数 (k_2):
 $1.55 \text{ L} \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$

修正した解析手法

1. 反応速度式:

化学反応式は以下の通りです。



反応速度式は前回と同様に、以下で表されます。

$$\frac{dx}{dt} = k_1(C_{A0} - x) \cdot (C_{B0} - x) - k_2 \cdot x^2$$

2. 収率(Y)の再定義:

ご指摘に基づき、MPC 収率(Y)を DMC の初期濃度 C_{A0} に対する定義に修正しました。

$$Y = \frac{C_A - C_{A0}}{C_{A0}} \rightarrow x = Y \cdot C_{A0}$$

これを時間で微分すると、 $\frac{dx}{dt} = C_{A0} \frac{dY}{dt}$ となります。これらの関係式を反応速度式に代入し、収率 Y に関する微分方程式を再導出しました。

$$\frac{dY}{dt} = k_1(1 - Y)(C_{B0} - Y \cdot C_{A0}) - k_2 Y^2 C_{A0}^2$$

この式を数値積分のモデルとして使用します。

3. 最適化手法の変更:

最適化手法として、よりカーブフィッティングに適した

`scipy.optimize.least_squares` を使用しました。この関数は、各データ点での残差（計算値と実験値の差）のベクトルを算出し、その二乗和を最小化することで最適なパラメータ (k_1, k_2) を探索します。

解析結果

再計算された速度定数を用いて計算した収率と、実験データを比較します。

反応時間 [h]	実験収率 [-]	計算収率 [-]
0.5	0.0110	0.0110
1.0	0.0215	0.0210
1.5	0.0304	0.0294
2.0	0.0361	0.0361
2.5	0.0403	0.0414
3.0	0.0442	0.0454
3.5	0.0449	0.0485
4.0	0.0461	0.0509
24.0	0.0474	0.0617

注: 24 時間後の平衡点において、モデルと実験値に若干の乖離が見られます。これは、長時間の反応では触媒の失活など、モデルでは考慮されていない他の要因が影響している可能性を示唆します。しかし、初期から中期（～4h）にかけては良好な一致を示しています。

グラフ（プロット表示）

以下のグラフは、提供する Python コードを実行すると表示され、結果の妥当性を視覚的に確認できます。

パリティプロット

実験値と計算値のプロットです。点が対角線上に乗るほど、モデルの当てはまりが良いことを示します。初期の反応領域では良好な一致が見られます。

収率の時間変化

実験データ（点）とモデルによる計算曲線（実線）の比較です。反応初期の挙動をよく捉えていることがわかります。

回答の中の速度定数の値や計算収率の値がおかしいのうまいっていないのかと思ったが、実際にコードを実行して得られた結果を見ると

```
=== 推定パラメータ ===
```

```
k1 = 1.156850e-06 L/mol/s
```

$k_2 = 4.545194 \times 10^{-4} \text{ L/mol/s}$

$C_{Ae} = 5.531593 \text{ mol/L}$

と回答の中に示されたものと違い妥当な値だった (Gemini の内部実行環境に起因する不具合があったと考えられる)。解析手法については丸投げしたため解析的な解を使うのではなく強引に数値積分をしていたが、当たり前ではあるが結果は妥当なものであった。ただはじめに出力した結果がうまくデータを説明できていないことの判断はできておらず限界があった。ここは人間の判断が必要だった。今回の不具合に対しては Python に関する知識が深くないと解決できないのでまだ AI が Python のどの関数を使うかの判断と解析した結果が妥当であるかの判断が十分でないことが分かった。

以上のように工学的に適切な指示を与えることで、Python に関する深い知識がなくても、解析結果およびコードを得ることが可能である。エンジニアとしては AI がツールであると割り切れれば、Python に関して深い知識がなくとも結果が得られればそれでよいといえる。解析の途中結果も出力され、解析方法と考え方も文書で残ることはこれまで必ずしもそこまで文書化していなかったことを考えればよいことのようにも思える。皆様は、このような AI との関わり方について、どのようにお考えになるでしょうか？

(日付 2025/6/17)