Project 3: Maleic Anhydride Production

Lev Gerasimov, lg726, Downing College

9th of February 2026

Word count: 1992

Contents

1 Summary

Three PFR reactor models for the production of Maleic Anhydride were explored: isothermal, adiabatic and with cooling. The isothermal case was used to approbate methods of yield and conversion analysis: butane conversion, yield and flowrate of Maleic Acid. Also J_{coef} was introduced. The adiabatic case was used to model heat accumulation, which accelerates side reactions and does not exist in isothermal case. In the final part of report the analysis of the cooling introduction was explored. The variability of coolant temperature was determined to be extremely sensitive parameter influencing the output. The other influential factor was inlet pressure and concentration of feed: higher feed concentration and pressure independently lead to higher conversion up to the full burnout. The introduction of the pressure drop into calculations lead to a narrow increase in productivity, while it is important in scaled manufacturing.

2 Introduction

The project is focusing on the process of the production of the maleic anhydride (MA) via the selective oxidation of the n-butane (butane) in the multi-tubular reactor with catalytic bed. The nature of the main and side combustion reactions is highly exothermic, hence, the performance is predominantly affected by the temperature.

The aim of the work is to model the reactor performance under multiple operational conditions: isothermal, adiabatic, with cooling; and analyse the effect on selectivity, yield, temperature profiles to consider the optimum values for every condition.

The reactor is assumed to be a plug flow reactor, using the kinetics provided in project brief, assuming excess of oxygen in the feed and the neglecting heat and mass transfer limitations between the gas and catalyst, simplifying model to homogeneous reactor.

Material and energy balances were solved numerically to obtain concentration and temperature profiles for the range of reactor volumes.

Firstly, the isothermal reactor model was assessed to identify the effects of range of temperatures on yield and selectivity.

Secondly, the adiabatic model was performing the effect of the heat release on the conversion and limitations of lack of cooling of the exothermic reactions.

Finally, the cooling system with molten salt was implemented to assess partially hypothetical case with no or ideal heat exchange. The cooling was assumed to perform as non-adiabatic system with constant cooling temperature.

The kinetic and thermodynamic equations were solved using numerical integration methods implemented in python([1]). The implementation methods are provided in the appendix.

3 Results and Discussion

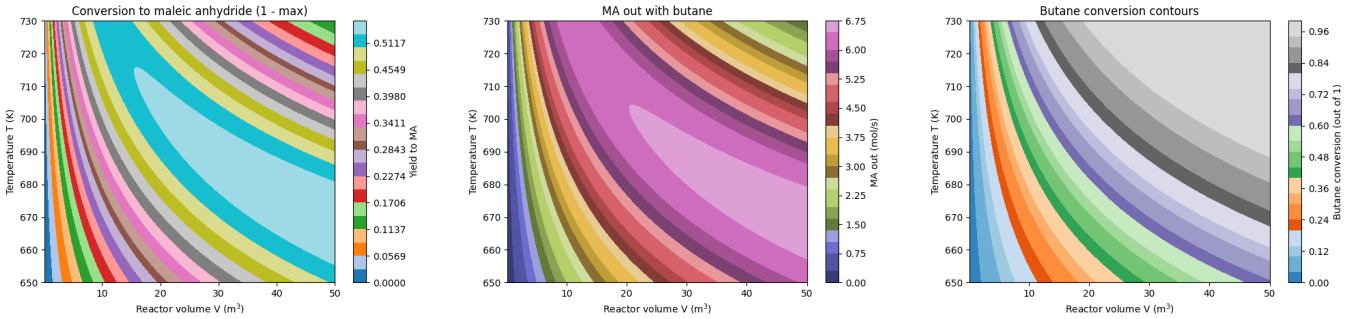
Every reactor model was using previously stated assumption: ideal PFR behaviour, excess of oxygen, neglect of the pressure drop and transport limitations.

3.1 Isothermal case

The main goal of the isothermal analysis is to consider the boundaries for the further, more complex simulations for the safe and operatable temperature profiles.

The modeling of the isothemal reactor at different volumes and temperatures provided following counter plots for the yield of MA (Fig. 1a), molar flow of MA (Fig. 1b), and depletion of butane (Fig. 1c).

For the counter plots the qualitative colourmap was used to separate the regions of different numbers not continuously, but discretely. The exactness of the maximum is not crucial due to operational uncertainties, so the quantifying colourmap considered as the nest.



(a) Yield of MA (b) Molar flow of MA (c) Depletion of butane

Figure 1: Isothermal reactor results

Figure 1a depicts that the maximal yield at the isothermal reactor happens at the large volumes and lower temperatures. Simultaneously, figure 1c the operation inlet temperature rise is directly related with the higher depletion of butane. Hence, the produced MA at higher temperatures burns down, loosing the product. Figure 1b shows more precise values of molar flow of MA from the reactor

The rough graph analysis based on the $\mathrm{MA_{out}}$ demonstrates that the realistic operational range for the reactor volume is from $35.0\mathrm{m}^3$ to $50.0\mathrm{m}^3$. The operational temperature of the reactor is enclosed between roughly 665K and 685K (392C and 412C).

However, from the butane depletion graph, the lower volume and lower temperature are considered to be more effective for future separation recycle (not considered in this project).

The selectivity ratio was used to narrow the operational region of volume to $0.1\mathrm{m}^3$ and temperature to 1K. However, yield could be low if only the selectivity is monitored, decreasing the rate as it is. Hence, new metric coefficient was introduced: J-coefficient.

J-coefficient is a multiplication of the selectivity by the yield (both on scale from 0 to 1).

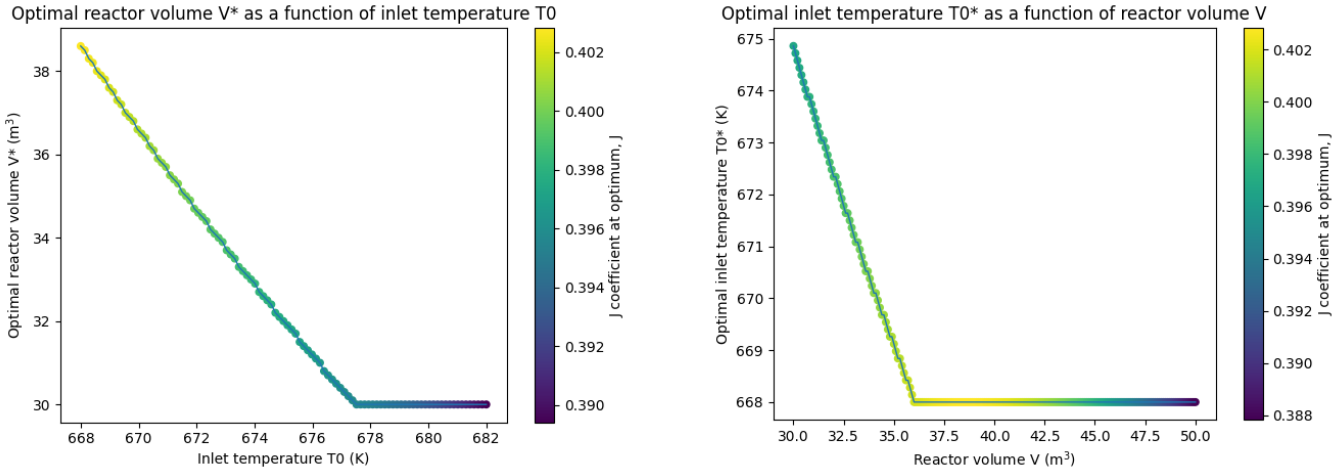
$$J_{\mathrm{MA,B}} = S_{\mathrm{MA,B}} \cdot Y_{\mathrm{MA,B}} = \frac{\mathrm{MA_{out}}}{\mathrm{B_{in}}} \cdot \frac{\mathrm{MA_{out}}}{\mathrm{B_{used}}} = \frac{\mathrm{MA_{out}^2}}{\mathrm{B_{in}} \cdot \mathrm{B_{used}}}$$

J-coefficient is used to de-emphasise the points with huge difference in yield and selectivity to focus on conditions with the most effective and rapid production of MA.

For the range of the volumes and temperatures were made two separate optimisations by the value of J-coefficient.

For every volume was found the temperature with highest J-coefficient (Fig. 2b).

For every temperature was found the volume with highest J-coefficient (Fig. 2a)



(a) Optimal volume of inlet temperature

(b) Optimal inlet temperature of volume

Figure 2: Optimal operating conditions based on $\mathrm{J_{coef}}$

Graphs show that the optimum for the higher temperatures lies in lower volumes and vice versa. Yet, the highest $\mathrm{J_{coef}}$ is found around the $37.5\mathrm{m}^3$ and 668K(395C).

Further it will be referred as Isothermal Optimum($37.5\mathrm{m}^3, 395\mathrm{C}$), yet it is only a dot in the area of optimum (operational region of volume to $0.1\mathrm{m}^3$ and temperature to 1K). The optimum point will never exist in reality due to uncertainties of the volume of the reactor and of the reading of the thermocouple. ([2])

The specific values for this point are demonstrated in Table 1.

Table 1: Isothermal results at $V = 37.5\,\mathrm{m}^3$ and $T = 668\,\mathrm{K}$.

Quantity	Value
MA out, MA_{out} (mol s^{-1})	6.387
Yield to MA, Y_{MA} (-)	0.532
Selectivity to MA, S_{MA} (-)	0.756
Butane conversion, X_B (-)	0.704
Objective, $J = Y_{\mathrm{MA}} S_{\mathrm{MA}}$ (-)	0.403

This Isothermal Optimum conditions have their operational risks and limitations.

First is the size of the reactor, which is considered huge for the usual PFR reactor. However, for many highly exothermic processes the volume of the reactor may be up to $50\mathrm{m}^3$. ([3]).

Second is the temperature. The optimal temperature found in model is significantly lower than the temperature of autoignition, hence, the this could be considered as a factor of low risk.

The isothermal model may be effective, yet it limits the realistic perception of the reactor modeling. The main reasons are:

1. **Rapid-adjusting heat consumption and heat distribution**
   Assuming that the reactor is isothermal means coolant taking just enough heat to keep the temperature same. However, the overall rate of the reactions and heat production varies dramatically, which is impossible to adjust in real time.

2. **Unrealistic temperature stability**
   High efficiency and high selectivity of the isothermal optimum are happening at the relatively low temperature with no risk of the autoignition. However, in real life heat will be absorbed firstly by the substances, increasing side-combustion pathways, decreasing the selectivity and yield.

Therefore, analysis of the system where the heat is not transferred to the surroundings may be helpful to overcome the limitations of the isothermal model.

The technical and predominantly hypothetical model to investigate will be the adiabatic model: it assumes that no heat transferred to the surrounding, allowing to model the temperature rise by heat accumulation, multiplying the effect of the side reactions.

3.2 Adiabatic case

The main goal of the adiabatic analysis is to consider the operational conditions when heat accumulation dominates and accelerating side-ways exothermic reactions. The rough analysis of the adiabatic conversion (Fig. 3) shows that the conversion happens predominantly at low volumes; thus, the primary analysis can be enclosed in the volume range from $0m^3$ to $3m^3$ and the temperature range from 660K and 680K (between 387°C and 407°C).

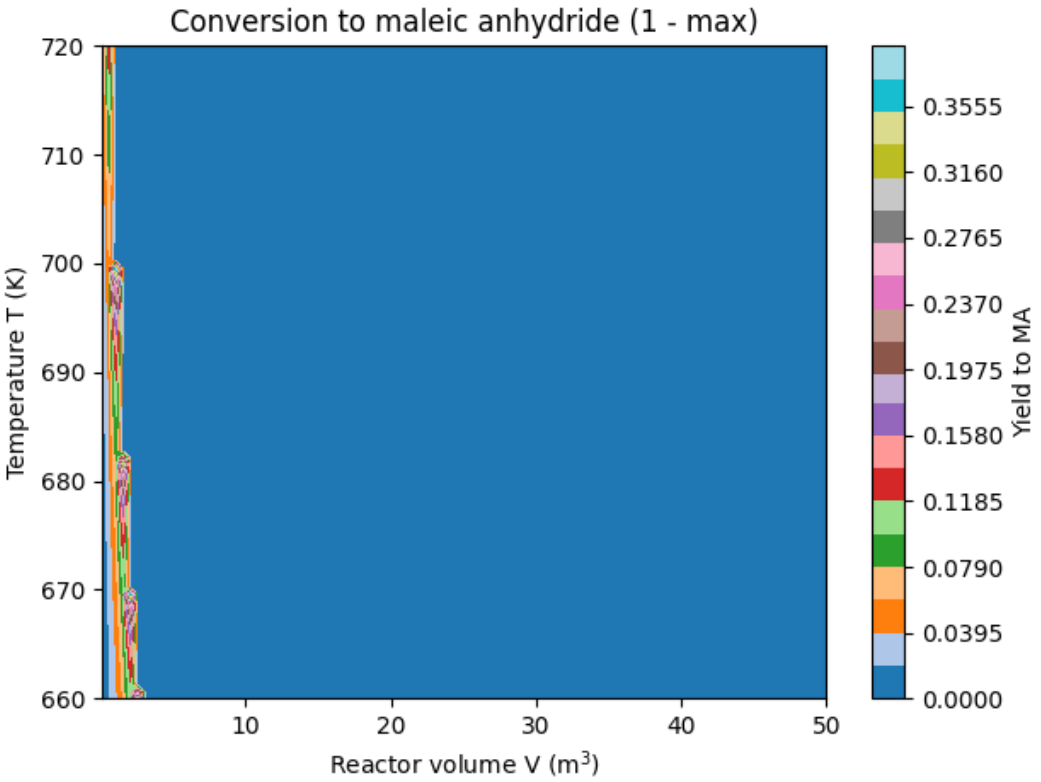


Figure 3: Adiabatic Yield of MA, rough

The following analysis of the yield (Fig. 4a) and molar flow (Fig. 4b) of MA and butane depletion profile (Fig. 4c) provides that the nature of adiabatic process is enormously sensitive to the inlet temperature and volume. Increasing volume by $0.5m^3$ and temperature by 0.5K may switch process from being highly effective to being totally wasteful, burning the butane totally.

Results for points (671.0 K, 2.0 m^3) and (671.5 K, 2.5 m^3) are the following:

Table 2: Adiabatic comparison at two operating points.

Quantity	$(T, V) = (671.0\,\text{K}, 2.0\,\text{m}^3)$	$(T, V) = (671.5\,\text{K}, 2.5\,\text{m}^3)$
MA_{out} (mol s^{-1})	3.44	9.23×10^{-3}
Y_{MA} (-)	0.286	7.69×10^{-4}
S_{MA} (-)	0.787	7.69×10^{-4}
X_B (-)	0.355	1.000
$J = Y_{\text{MA}} S_{\text{MA}}$ (-)	0.226	5.92×10^{-7}

The border of these conditions is identified as the border for null yield and high yield of MA, which is exactly the border for the complete depletion of butane.

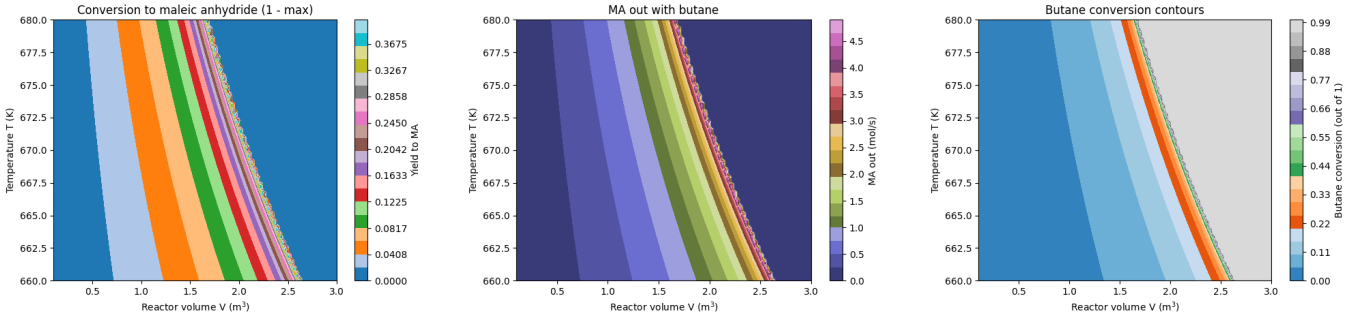
Thus, the uncertainties of monitoring of the volume of reactor and the temperature inlet are serious threats to the effectiveness of production and ought to be considered separately.

The k-type thermocouple of the 1 class of precision have an uncertainty of ±1.5°C. ([2])

The ASME Section VIII manufacturing standards ([3]) allows for reactors of volume from 1000 to 5000 liters (from $1.0m^3$ to $5.0m^3$) have uncertainty of ±1.0%. As our operational conditions will be around $2.5m^3$, the uncertainty of the volume is considered $\pm 0.025m^3$.

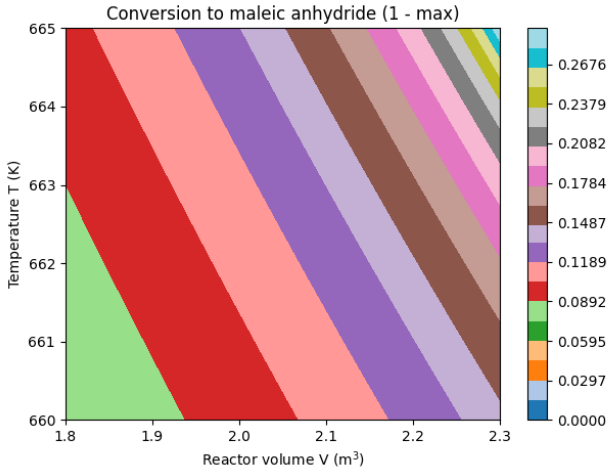
The other reason to consider is whether or not full combustion in the square of uncertainty. The reason for the saving of the butane in isothermal case is also valid here; hence, the upper limit of temperature and volume uncertainty should be the region of the most efficiency (which borders with the area of full depletion of butane and null yield).

The operational temperature and volume considered from the Fig. 4b are 663K (390°C) and $2.05m^3$. The operational risks at this temperature are minimal as the point of autoignition is unallowed by the temperature sensors to

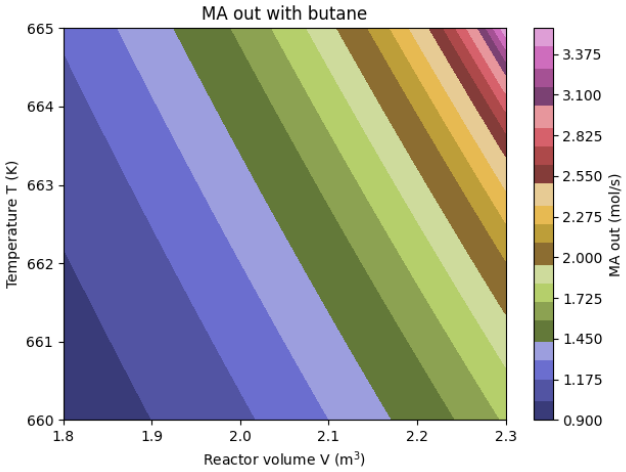


(a) Yield of MA (b) Molar flow of MA (c) Depletion of butane

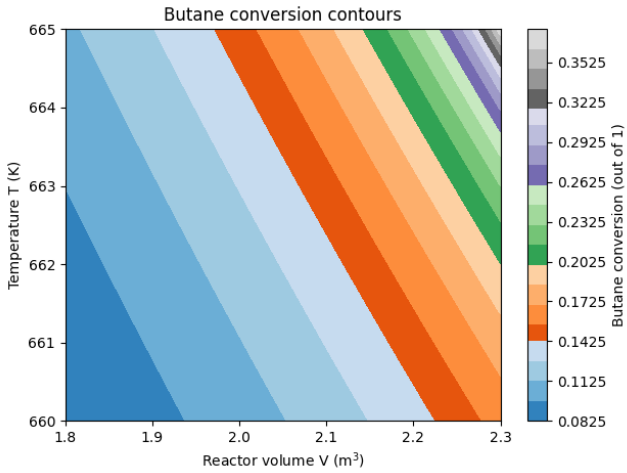
Figure 4: Adiabatic analysis of the reactor, narrowed



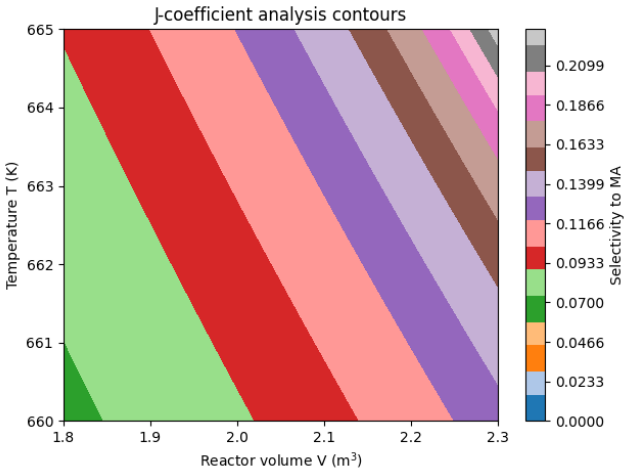
(a) Yield of MA, precise



(b) Flowrate of MA, precise



(c) Butane depletion, precise



(d) Adiabatic J_{coef}, precise

Figure 5: Adiabatic analysis of the operational unit with temperature at 663K (390°C) and operational volume 2.05m^3

be reached. Moreover, moderate J_{coef} (Fig. 5d) with low butane depletion (Fig. 5c) performs high selectivity of the operational conditions, allowing to implement the modeled reactor in the recycle chain to minimize wasting of the butane to the carbon oxides and steam.

On the other hand, the expected flowrate of MA (Fig. 5b) and yield (Fig. 5a) are considered to be low for the non-recyclce operational unit. This could not be neglected due to the narrowity of the high-efficiency region of the adiabatic case, which could be taking into account only with either low yield and high selectivity if temperature drops or with null yield and full butane depletion if temperature rises.

In reality, the adiabatic process is impossible unless the reactor is covered in thick layer the high-quality insulator as ceramics. Allowing heat to be transfered from the accumulated regions to the coolent may prevent side reactions and increase the high-efficiency region of the reactor, allowing to consider higher-yield approach rather than fuel-saving one.

3.3 Coolant case

The purpose of the analysis of the reactor with coolant is to model behaviour closer to real-life operation, where the temperature is not constant, yet heat is not perpetually accumulated due to continuous heat removal.

The analysis is divided into four parts.

Part *a* considers the operating conditions specified in the project brief.

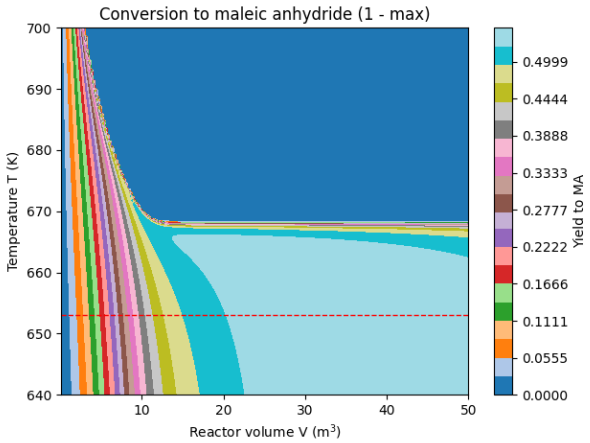
Part *b* focuses on determining the effect of varying both the inlet and coolant temperatures.

Part *c* analyses the effect of changing the inlet pressure and the feed concentration of butane.

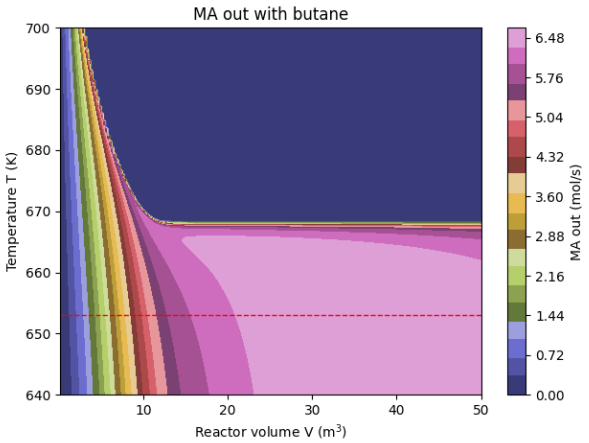
Part *d* presents the effect of introducing pressure drop into the simulation.

The tubes used in the coolant system in the following analysis have a length of 6 m ([4]).

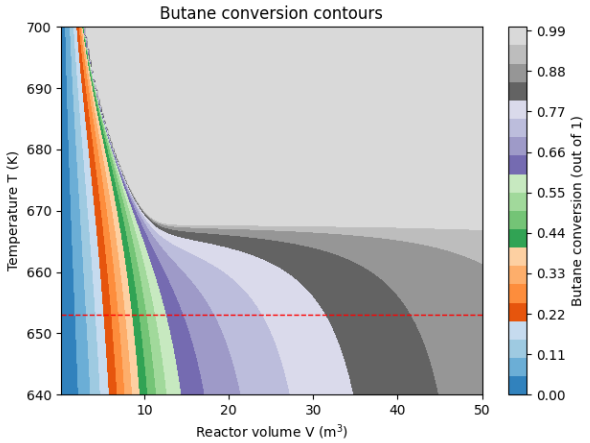
3.3.1 Part a: $T_{inlet} = T_{coolant} = 380°C$



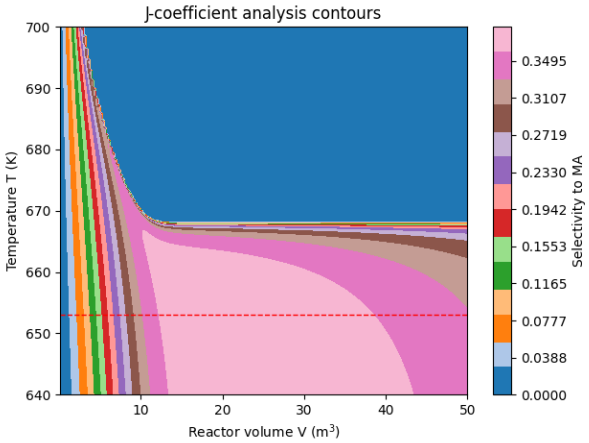
(a) Yield of MA



(b) Flow rate of MA



(c) Butane conversion



(d) J-coefficient

Figure 6: Cooling system analysis at $T_{coolant} = 380°C$. The red dashed line indicates $T_{inlet} = T_{coolant}$.

The yield of MA (Fig. 6a) and the flow rate of MA (Fig. 6b) show that, under the specified operating conditions, increasing the reactor volume has a diminishing effect on the targeted output once the volume exceeds approximately 22 m^3. This value can therefore be considered as a lower boundary for the target reactor volume.

However, the J-coefficient analysis (Fig. 6d) indicates that at larger reactor volumes the selectivity decreases while the yield remains nearly constant. This implies that an increasing fraction of butane is consumed in side reactions rather than converted to maleic anhydride. This behaviour is supported by the contour plot of butane conversion (Fig. 6c), which exceeds the 80% conversion threshold at high volumes. Consequently, further increases in reactor volume become inefficient and define an upper bound for the optimal operating range at this temperature.

The behaviour of the MA yield and flow rate at the temperature specified in the project brief is significantly more favourable compared to higher temperatures. Moreover, the temperature profiles in this region resemble quasi-isothermal behaviour rather than adiabatic operation. This effect is achieved due to the large heat exchange area of the cooling system,

$$\text{Area} = N\pi DL = 10000 \cdot \pi \cdot 0.025 \cdot 6 = 4712.4\,\text{m}^2.$$

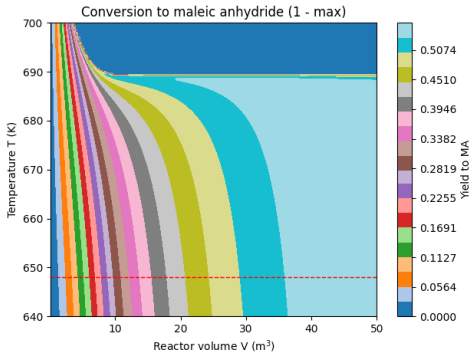
At higher temperatures (above approximately 670 K or 397°C), the reactor exhibits a more pronounced adiabatic

temperature profile. In this regime, the cooling system becomes less effective in compensating for heat accumulation, and the reactor behaviour increasingly approaches that of an adiabatic system.

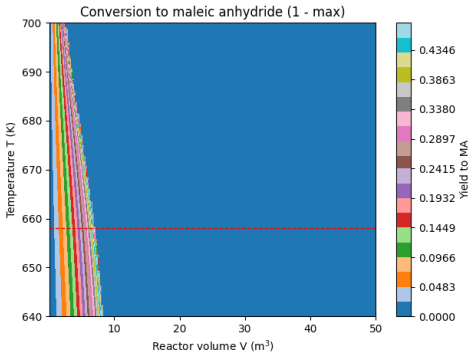
3.3.2 **Part b:** Range of T_{inlet} and $T_{coolant}$

The effect of varying the inlet temperature was discussed in the previous part by comparing temperature profiles at different reactor volumes. The analysis showed that temperatures above 670 K result in predominantly adiabatic behaviour, while at lower temperatures the system behaves closer to isothermal operation.

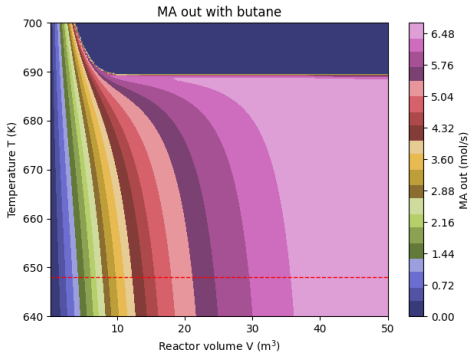
For further analysis, the coolant temperature was increased to 385°C and decreased to 375°C. The condition $T_{\text{inlet}} = T_{\text{coolant}}$ is indicated in the figures by the red dashed line.



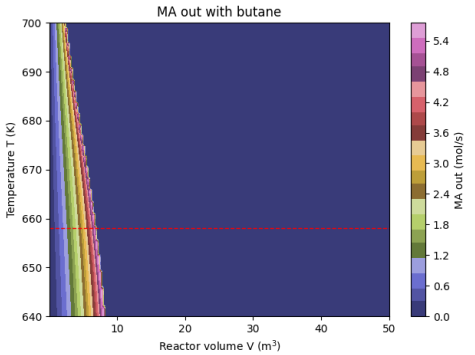
(a) Yield of MA (375°C)



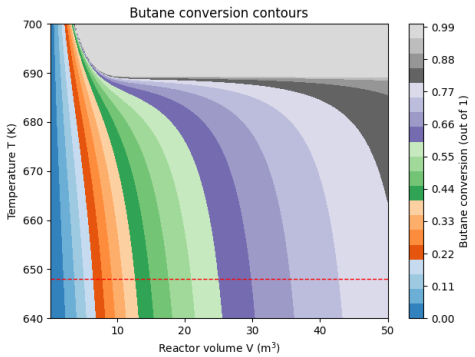
(b) Yield of MA (385°C)



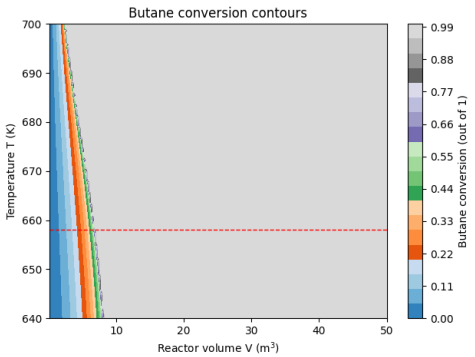
(c) MA flow rate (375°C)



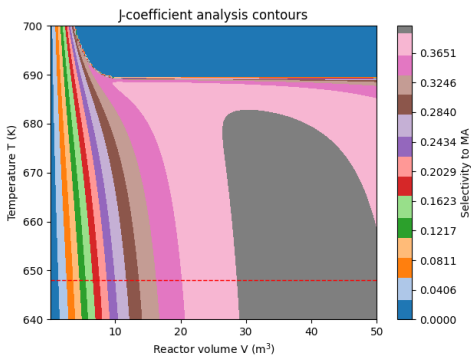
(d) MA flow rate (385°C)



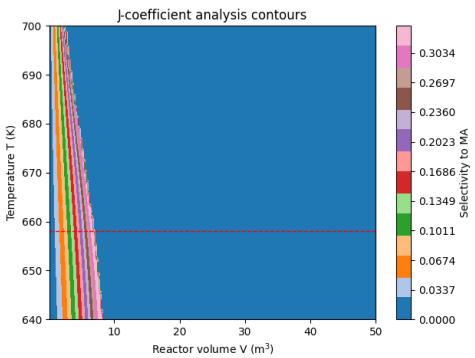
(e) Butane conversion (375°C)



(f) Butane conversion (385°C)



(g) J-coefficient (375°C)



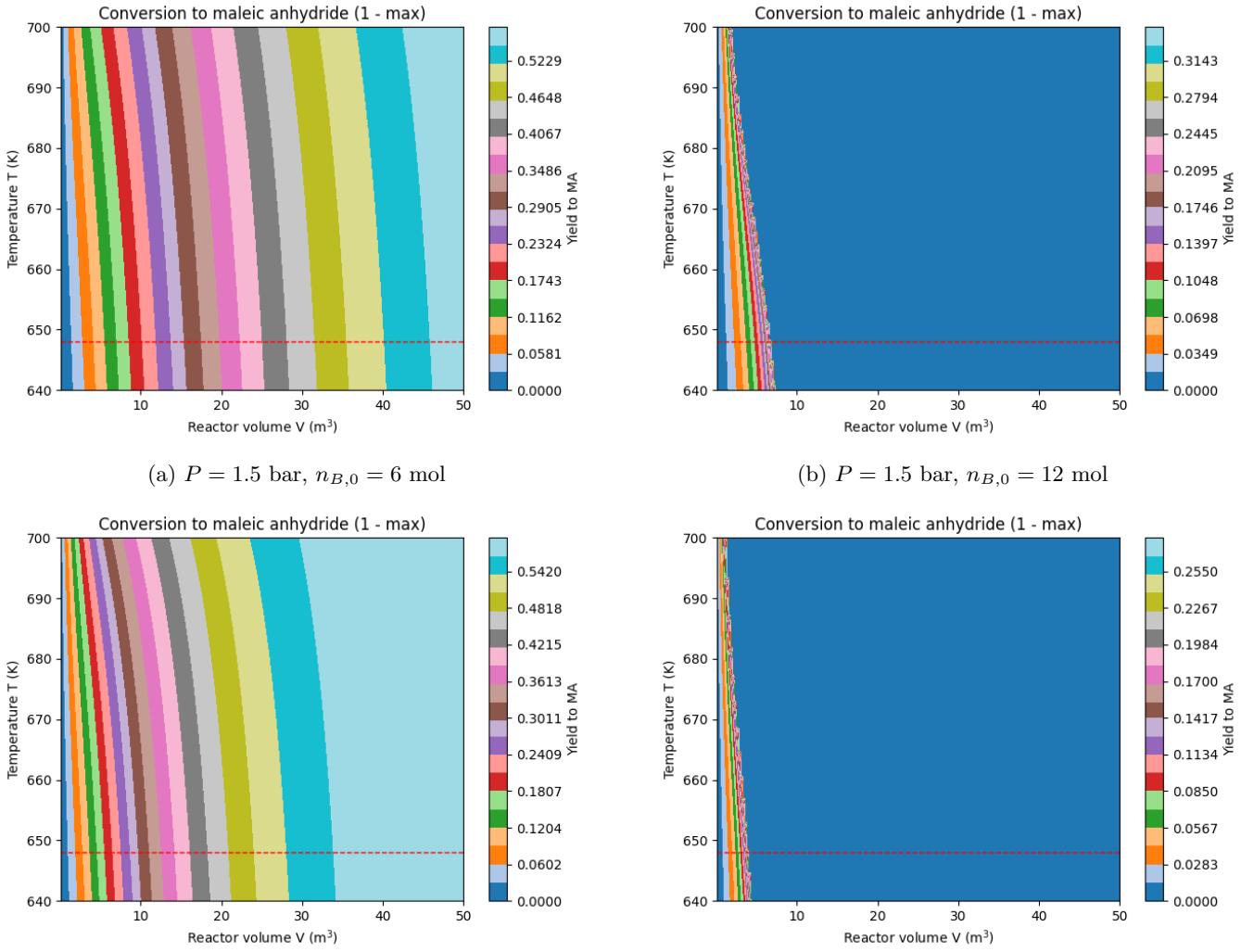
(h) J-coefficient (385°C)

Figure 7: Cooling-system contour comparison for two coolant temperatures.

The variation of the coolant temperature significantly affects the MA yield, MA flow rate, butane conversion, and J-coefficient profiles (Fig. 7). Increasing the coolant temperature to 385°C substantially narrows the operational reactor volume range at the condition $T_{\text{inlet}} = T_{\text{coolant}}$, with the upper feasible volume reduced to approximately $8\,\text{m}^3$. In contrast, at 375°C the corresponding operational volume range extends from approximately $30\,\text{m}^3$ to $50\,\text{m}^3$.

A higher coolant temperature reduces the effective temperature driving force for heat removal and therefore promotes adiabatic-like behaviour. Consequently, increasing either the inlet temperature or the coolant temperature shifts the reactor operation toward the adiabatic regime.

3.3.3 **Part c:** Effect of changing inlet pressure and concentration of butane



(a) $P = 1.5$ bar, $n_{B,0} = 6$ mol

(b) $P = 1.5$ bar, $n_{B,0} = 12$ mol

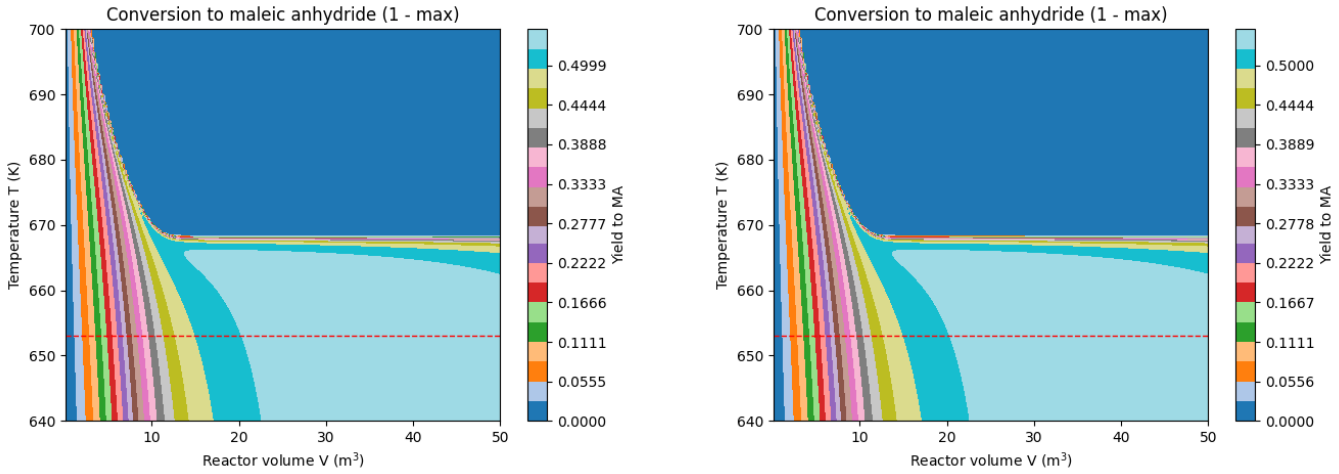
(c) $P = 2.5$ bar, $n_{B,0} = 6$ mol

(d) $P = 2.5$ bar, $n_{B,0} = 24$ mol

Figure 8: Yield contours for different inlet pressures and initial butane amounts.

The yield contours presented in Fig. 8 show that lowering the inlet butane concentration leads to an increase in the yield of maleic anhydride, while increasing the inlet pressure enhances the overall conversion. At higher inlet concentrations, an increase in pressure shifts the boundary of near-complete butane conversion toward lower reactor volumes, reflecting higher reaction rates due to increased partial pressures.

3.3.4 **Part d:** Effect of introduction of pressure drop



(a) Constant-pressure model

(b) Model with pressure drop

Figure 9: Effect of including pressure drop on MA yield contours.

The comparison of simulations with and without pressure drop (Fig. 9) shows that accounting for pressure losses along the reactor slightly increases the reactor volume required to reach a given level of conversion. This effect arises from the gradual reduction of partial pressures along the reactor length, which lowers reaction rates compared to the constant-pressure assumption.

The differences in the maximum attainable conversion between the two models are small, on the order of 0.2%. Although modest, such differences may become significant when extrapolated to industrial-scale production rates.

4 Recommendations

The production of MA is effective in a non-adiabatic multitubular reactor with molten-salt cooling under the following recommended (nominal) design and operating conditions:

1. **Nominal temperatures:** $T_{inlet} \approx T_{coolant} = 380\,^{\circ}C$ (set-point; local deviations are expected in operation).
2. **Reactor size:** $V_{reactor} \approx 30\,m^3$ (rounded design value within the high-performance operating region, rather than a sharp optimum).
3. **Temperature monitoring/control:** K-type thermocouples, Class 1 accuracy ($\pm 1.5\,^{\circ}C$), used to define a practical safety/control margin.
4. **Tube bundle geometry:** $D = 25\,mm$, $L = 6\,m$ (standard manufacturable/transportable tube length).
5. **Number of tubes:** $N \approx 10{,}000$ (order-of-magnitude estimate based on required heat-transfer area).
6. **Coolant configuration (assumption):** molten salt operated with a controlled inlet temperature and sufficiently high flow rate to maintain near-uniform coolant-side conditions along the reactor.
7. **Operating window recommendation:** the reactor should be operated within a temperature–volume window around the nominal point, rather than targeting a single exact (T, V), due to strong sensitivity and instrumentation/manufacturing uncertainties.

5 Reference

1. Barrie, P. (n.d.). *CEBT Part IB Project 3: Reaction Engineering — Production of Maleic Anhydride*. University of Cambridge.

2. Farr, J. R. and Jawad, M. H. (2010). *Guidebook for the Design of ASME Section VIII Pressure Vessels*. New York, NY: ASME.

3. Green, D. W. and Perry, R. H. (2008). *Perry's Chemical Engineers' Handbook*, 8th ed. New York: McGraw–Hill.

4. Rogoff, G. L. (1973). Thermocouple structure for measurements of surface temperature. *Review of Scientific Instruments*, 44(5), pp. 654–655. doi: `https://doi.org/10.1063/1.1686214`

6 Appendix

A Code

A.1 Isothermal Project 3.py

```
# Lev Gerasimov's Project 3 code
# Right now I approximate the isothermal situation of the reactor of maleic ahidryde
# January 2026

# 3 reactions in gas phase
# C4H10 + 3.5_O2 = C4H2O3 + 5_H2O
# C4H2O3 + 2_O2 = 2_CO + 2_CO2 + H2O
# C4H10 + 5_O2 = 3_CO + CO2 + 5_H2O
# list of species and their points below
# 0 - C4H10
# 1 - O2
# 2 - C4H2O3
# 3 - H2O
# 4 - CO
# 5 - CO2
# 6 - N2 (in air)


import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# number of species
nspec = 7

# number of reactions
nrec = 3

# stoichiometric coefficients converted into matrix
stoic_1 = np.array([-1, -3.5, 1, 4, 0, 0, 0])
stoic_2 = np.array([0, -2, -1, 1, 2, 2, 0])
stoic_3 = np.array([-1, -5, 0, 5, 3, 1, 0])

stoic_matrix = np.stack([stoic_1, stoic_2, stoic_3])

# Arrhenius coefficients and activation energy for three reactions
A_matrix = np.array([
    20.0, # Reaction 1: selective oxidation to maleic anhydride
    4.0, # Reaction 2: oxidation of maleic anhydride
    0.7 # Reaction 3: combustion of n-butane
])

E_act_matrix = np.array([
    95e3, # Reaction 1
    105e3, # Reaction 2
    125e3 # Reaction 3
])


def k_calc(A_matrix, E_act_matrix, T):
    T1 = T
    k_new = A_matrix * np.exp(-(E_act_matrix / 8.3145) * (1.0 / T1 - 1.0 / 673.15))
    return k_new


# rate coefficients for the reactions
KMA = 10
KB = 22
KS = 2

EPS = 1e-12

```

```
# Enthalpies of formation at 298 K (in base SI units)
delHf = np.array([
    -125.6e3, # n-Butane, kJ/mol -> J/mol
    0.0, # Oxygen
    -398.3e3, # Maleic anhydride (vapour)
    -241.8e3, # Water (steam)
    -110.5e3, # Carbon monoxide
    -393.5e3, # Carbon dioxide
    0.0 # Nitrogen
])

# Heat capacity coefficients (in base SI units)
# Cp_m(T) = a + b*T + c*T^2
# Each Cp_paramX is [a, b, c] for species X (see species numbering above).

Cp_param0 = np.array([3.79, 358e-3, -137e-6]) # 0 - n-Butane
Cp_param1 = np.array([26.5, 9.4e-3, -0.3e-6]) # 1 - O2
Cp_param2 = np.array([14.4, 294e-3, -138e-6]) # 2 - Maleic anhydride (vapour)
Cp_param3 = np.array([32.0, 3.2e-3, 6.6e-6]) # 3 - H2O (steam)
Cp_param4 = np.array([28.9, -1.2e-3, 6.3e-6]) # 4 - CO
Cp_param5 = np.array([22.3, 58.3e-3, -27.4e-6]) # 5 - CO2
Cp_param6 = np.array([29.4, -3.2e-3, 7.3e-6]) # 6 - N2

Cp_param_matrix = np.stack([
    Cp_param0,
    Cp_param1,
    Cp_param2,
    Cp_param3,
    Cp_param4,
    Cp_param5,
    Cp_param6
])

# functions to calculate the heat capacities of species at temperature T


def Cp_calc(nspec, Cp_param_matrix, T):
    T_func = np.array([1, T, T**2])
    Cp_val = Cp_param_matrix @ T_func
    return Cp_val


# function to calculate the enthalpy of reaction at temperature T


def delHreac(stoic_matrix, delHf, cp_param_matrix, T):
    delHf298 = stoic_matrix @ delHf
    delCP = stoic_matrix @ cp_param_matrix
    T_func = np.array([T - 298.0, (T**2 - 298.0**2) / 2.0, (T**3 - 298.0**3) / 3.0])
    delH_reac = delHf298 + delCP @ T_func
    return delH_reac


# function to model the isothermal operation


def rhs_isothermal(V, xi):
    P = P0
    ndot = ndot0 + xi @ stoic_matrix
    ndot = np.maximum(ndot, 0.0)

    ndot_tot = float(ndot.sum())
    if ndot_tot <= EPS:
        return np.zeros(3)

    y = ndot / ndot_tot
    y = np.maximum(y, 0.0)
    y_sum = float(y.sum())
    if y_sum <= EPS:
```

```
        return np.zeros(3)
    y = y / y_sum

    P_part = y * P
    P_part = np.maximum(P_part, 0.0)

    PB = P_part[0]
    PMA = P_part[2]
    PS = P_part[3]

    k1, k2, k3 = k_calc(A_matrix, E_act_matrix, T0) # constant T0 for isothermal
    r1 = k1 * PB / (1 + KB * PB + KMA * PMA + KS * PS)
    r2 = k2 * PMA
    r3 = k3 * PB

    return np.array([r1, r2, r3])


# set conditions at inlet (in base SI units)
P0 = 2.0
ndot0 = np.array([12.0, 137.48, 0.0, 0.0, 0.0, 0.0, 517.19])

# specify number of data points for displayed solution
ndata = 500

# ranges for isothermal analysis
Ts = np.linspace(668, 678, 101) # K
Vs = np.linspace(0.1, 50, 101) # m^3

MA_grid = np.zeros((len(Ts), len(Vs)))
Y_grid = np.zeros_like(MA_grid)
S_grid = np.zeros_like(MA_grid)
XB_grid = np.zeros_like(MA_grid)

for i, T0 in enumerate(Ts):
    for j, Vreactor in enumerate(Vs):
        V_span = (0.0, float(Vreactor))
        xi0 = np.zeros(nrec)

        sol = solve_ivp(rhs_isothermal, V_span, xi0, method="BDF", rtol=1e-6, atol=1e-8)

        if not sol.success or sol.y.size == 0:
            MA_grid[i, j] = np.nan
            Y_grid[i, j] = np.nan
            S_grid[i, j] = np.nan
            XB_grid[i, j] = np.nan
            continue

        xi_end = sol.y[:, -1]

        MA_out = float(xi_end[0] - xi_end[1])
        B_cons = float(xi_end[0] + xi_end[2])

        if B_cons <= 0:
            S_MA = 0.0
            Y_MA = 0.0
            X_B = 0.0
        else:
            S_MA = MA_out / B_cons
            Y_MA = MA_out / float(ndot0[0])
            X_B = B_cons / float(ndot0[0])

        MA_grid[i, j] = MA_out
        Y_grid[i, j] = Y_MA
        S_grid[i, j] = S_MA
        XB_grid[i, j] = X_B

# combined objective: yield  selectivity
J_grid = Y_grid * S_grid
```

```
201
202 # contour plot mesh
203 V_mesh, T_mesh = np.meshgrid(Vs, Ts)
204
205 # 1) Contour plot for yield to MA
206 plt.figure()
207 levels_y = np.linspace(0.0, np.nanmax(Y_grid), 120)
208 cf = plt.contourf(V_mesh, T_mesh, Y_grid, levels=levels_y, cmap="tab20", vmin=0.0, vmax=np.nanmax(Y_grid
    ))
209 plt.colorbar(cf, label="Yield␣to␣MA")
210 plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
211 plt.ylabel("Temperature␣T␣(K)")
212 plt.title("Conversion␣to␣maleic␣anhydride␣(1␣-␣max)")
213 plt.tight_layout()
214 plt.show()
215
216 # 2) MA production rate
217 plt.figure()
218 cf = plt.contourf(V_mesh, T_mesh, MA_grid, levels=120, cmap="tab20b")
219 plt.colorbar(cf, label="MA␣out␣(mol/s)")
220 plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
221 plt.ylabel("Temperature␣T␣(K)")
222 plt.title("MA␣out␣with␣butane")
223 plt.tight_layout()
224 plt.show()
225
226 # 3) Butane conversion
227 plt.figure()
228 cf = plt.contourf(V_mesh, T_mesh, XB_grid, levels=120, cmap="tab20c")
229 plt.colorbar(cf, label="Butane␣conversion␣(out␣of␣1)")
230 plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
231 plt.ylabel("Temperature␣T␣(K)")
232 plt.title("Butane␣conversion␣contours")
233 plt.tight_layout()
234 plt.show()
235
236 # 4) Selectivity
237 plt.figure()
238 levels_s = np.linspace(0.0, np.nanmax(S_grid), 120)
239 cf = plt.contourf(V_mesh, T_mesh, S_grid, levels=levels_s, cmap="tab20", vmin=0.0, vmax=np.nanmax(S_grid
    ))
240 plt.colorbar(cf, label="Selectivity␣to␣MA")
241 plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
242 plt.ylabel("Temperature␣T␣(K)")
243 plt.title("Selectivity␣to␣MA␣with␣butane")
244 plt.tight_layout()
245 plt.show()
246
247 # 5) J-coefficient
248 plt.figure()
249 levels_j = np.linspace(0.0, np.nanmax(J_grid), 120)
250 cf = plt.contourf(V_mesh, T_mesh, J_grid, levels=levels_j, cmap="tab20", vmin=0.0, vmax=np.nanmax(J_grid
    ))
251 plt.colorbar(cf, label="J␣coefficient")
252 plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
253 plt.ylabel("Temperature␣T␣(K)")
254 plt.title("J-coefficient␣analysis␣contours")
255 plt.tight_layout()
256 plt.show()
257
258 # -------------------------
259 # Optimal operating curves (reduce 2D grid to 1D optimum slices)
260 # -------------------------
261
262 # 1) Best V for each T0
263 V_opt_per_T = np.full(len(Ts), np.nan)
264 J_opt_per_T = np.full(len(Ts), np.nan)
265 for i in range(len(Ts)):
266     row = J_grid[i, :]
```

```
    if np.all(np.isnan(row)):
        continue
    j_best = int(np.nanargmax(row))
    V_opt_per_T[i] = Vs[j_best]
    J_opt_per_T[i] = row[j_best]

plt.figure()
sc = plt.scatter(Ts, V_opt_per_T, c=J_opt_per_T, s=22)
plt.plot(Ts, V_opt_per_T, linewidth=1)
plt.colorbar(sc, label="J coefficient at optimum, J")
plt.xlabel("Inlet temperature T0 (K)")
plt.ylabel("Optimal reactor volume V* (m$^3$)")
plt.title("Optimal reactor volume V* as a function of inlet temperature T0")
plt.tight_layout()
plt.show()

# 2) Best T0 for each V
T_opt_per_V = np.full(len(Vs), np.nan)
J_opt_per_V = np.full(len(Vs), np.nan)
for j in range(len(Vs)):
    col = J_grid[:, j]
    if np.all(np.isnan(col)):
        continue
    i_best = int(np.nanargmax(col))
    T_opt_per_V[j] = Ts[i_best]
    J_opt_per_V[j] = col[i_best]

plt.figure()
sc = plt.scatter(Vs, T_opt_per_V, c=J_opt_per_V, s=22)
plt.plot(Vs, T_opt_per_V, linewidth=1)
plt.colorbar(sc, label="J coefficient at optimum, J")
plt.xlabel("Reactor volume V (m$^3$)")
plt.ylabel("Optimal inlet temperature T0* (K)")
plt.title("Optimal inlet temperature T0* as a function of reactor volume V")
plt.tight_layout()
plt.show()

# --------------------------
# Print all data for a specific grid point (V=37.5 m^3, T=668 K)
# --------------------------

V_target = 37.5
T_target = 668.0

i_candidates = np.where(np.isclose(Ts, T_target, atol=1e-9))[0]
j_candidates = np.where(np.isclose(Vs, V_target, atol=1e-9))[0]

if len(i_candidates) == 0:
    raise ValueError(f"T_target={T_target} not found in Ts grid.")
if len(j_candidates) == 0:
    raise ValueError(f"V_target={V_target} not found in Vs grid.")

i = int(i_candidates[0])
j = int(j_candidates[0])

print("\n==============================")
print("Point data for (V=37.5 m^3, T=668 K)")
print("==============================")
print(f"Grid indices: i(T)={i}, j(V)={j}")
print(f"T_grid={Ts[i]:.6f} K, V_grid={Vs[j]:.6f} m^3\n")

print("From precomputed grids:")
print(f"MA_out (mol/s)           = {MA_grid[i, j]:.12g}")
print(f"Yield to MA (Y_MA)       = {Y_grid[i, j]:.12g}")
print(f"Selectivity to MA (S_MA) = {S_grid[i, j]:.12g}")
print(f"Butane conversion (X_B)  = {XB_grid[i, j]:.12g}")
print(f"J = Y*S                  = {J_grid[i, j]:.12g}")
```

A.2 Adiabatic Project 3.py

```
# Lev Gerasimov's Project 3 code
# Right now I approximate the adiabatic situation of the reactor of maleic acid
# January 2026

# 3 reactions in gas phase
# C4H10 + 3.5_O2 = C4H2O3 + 5_H2O
# C4H2O3 + 2_O2 = 2_CO + 2_CO2 + H2O
# C4H10 + 5_O2 = 3_CO + CO2 + 5_H2O
# list of species and their points below
# 0 - C4H10
# 1 - O2
# 2 - C4H2O3
# 3 - H2O
# 4 - CO
# 5 - CO2
# 6 - N2 (in air)


import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# number of species
nspec = 7

# number of reactions
nrec = 3

# stoichiometric coefficients converted into matrix
stoic_1 = np.array([-1, -3.5, 1, 4, 0, 0, 0])
stoic_2 = np.array([0, -2, -1, 1, 2, 2, 0])
stoic_3 = np.array([-1, -5, 0, 5, 3, 1, 0])

stoic_matrix = np.stack([stoic_1, stoic_2, stoic_3])

#Arrhenius coefficients and activation energy for three reactions
A_matrix = np.array([
    20.0, # Reaction 1: selective oxidation to maleic anhydride
    4.0, # Reaction 2: oxidation of maleic anhydride
    0.7 # Reaction 3: combustion of n-butane
])

E_act_matrix = np.array([
    95e3, # Reaction 1
    105e3, # Reaction 2
    125e3 # Reaction 3
])

def k_calc(A_matrix, E_act_matrix, T):
    T1 = T
    k_new = A_matrix * np.exp(-(E_act_matrix / 8.3145) * (1.0 / T1 - 1.0 / 673.15))
    return k_new
# rate coefficients for the reactions
KMA = 10
KB = 22
KS = 2

EPS = 1e-12

# Enthalpies of formation at 298 K (in base SI units)
delHf = np.array([
    -125.6e3, # n-Butane, kJ/mol -> J/mol
    0.0, # Oxygen
    -398.3e3, # Maleic anhydride (vapour)
    -241.8e3, # Water (steam)
    -110.5e3, # Carbon monoxide
    -393.5e3, # Carbon dioxide
```

```
    0.0 # Nitrogen
])

# Heat capacity coefficients (in base SI units)
# Cp_m(T) = a + b*T + c*T^2
# Each Cp_paramX is [a, b, c] for species X (see species numbering above).

Cp_param0 = np.array([3.79, 358e-3, -137e-6]) # 0 - n-Butane
Cp_param1 = np.array([26.5, 9.4e-3, -0.3e-6]) # 1 - O2
Cp_param2 = np.array([14.4, 294e-3, -138e-6]) # 2 - Maleic anhydride (vapour)
Cp_param3 = np.array([32.0, 3.2e-3, 6.6e-6]) # 3 - H2O (steam)
Cp_param4 = np.array([28.9, -1.2e-3, 6.3e-6]) # 4 - CO
Cp_param5 = np.array([22.3, 58.3e-3, -27.4e-6]) # 5 - CO2
Cp_param6 = np.array([29.4, -3.2e-3, 7.3e-6]) # 6 - N2

Cp_param_matrix = np.stack([
    Cp_param0,
    Cp_param1,
    Cp_param2,
    Cp_param3,
    Cp_param4,
    Cp_param5,
    Cp_param6
])

# functions to calculate the heat capacities of species at temperature T

def Cp_calc(nspec, Cp_param_matrix, T):
    T_func = np.array([1, T, T**2])
    Cp_calc = Cp_param_matrix @ T_func
    return Cp_calc

# function to calculate the enthalpy of species at temperature T

def delHreac(stoic_matrix, delHf, cp_param_matrix, T):
    delHf298 = stoic_matrix @ delHf
    delCP = stoic_matrix@ cp_param_matrix
    T_func = np.array([T - 298.0, (T**2 - 298.0**2)/2.0, (T**3-298.0**3)/3.0])
    delHreac = delHf298 + delCP @ T_func
    return delHreac

#function to model the adiabatic operation

def rhs_adiabatic(V, z):
    P = P0
    xi = z[0:3]
    T = z[-1]

    ndot = ndot0 + xi @ stoic_matrix

    # --- Physical / numerical guards ---
    # Clamp tiny negatives from numerical noise to zero.
    # Large negatives indicate the solver stepped past depletion; events should prevent this.
    if np.any(ndot < -1e-6):
        return np.zeros(4)
    ndot = np.maximum(ndot, 0.0)

    ndot_tot = float(ndot.sum())
    if ndot_tot <= EPS:
        return np.zeros(4)

    y = ndot / ndot_tot
    # Guard against any numerical drift in mole fractions
    y = np.maximum(y, 0.0)
    y_sum = float(y.sum())
    if y_sum <= EPS:
        return np.zeros(4)
    y = y / y_sum

```

```
    P_part = y * P
    # Ensure non-negative partial pressures
    P_part = np.maximum(P_part, 0.0)

    PB = P_part[0]
    PMA = P_part[2]
    PS = P_part[3]

    k1, k2, k3 = k_calc(A_matrix, E_act_matrix, T) # temperature-dependent kinetics
    r1 = k1 * PB / (1 + KB * PB + KMA * PMA + KS * PS)
    r2 = k2 * PMA
    r3 = k3 * PB

    R_tot = np.array([r1, r2, r3])

    Cp = Cp_calc(nspec, Cp_param_matrix, T)
    Cpdot = float(ndot @ Cp)
    if Cpdot <= EPS or not np.isfinite(Cpdot):
        return np.zeros(4)

    Hreact = delHreac(stoic_matrix, delHf, Cp_param_matrix, T)
    dT = -float(np.dot(Hreact, R_tot)) / Cpdot
    if not np.isfinite(dT):
        return np.zeros(4)

    ret = np.array([r1, r2, r3, dT])

    return (ret)

# -------------------------
# Event functions (stop when a key reactant is depleted)
# -------------------------

def event_butane_depleted(V, z):
    # Stop integration when butane molar flow reaches zero
    xi = z[0:3]
    ndot = ndot0 + xi @ stoic_matrix
    return float(ndot[0]) # C4H10

event_butane_depleted.terminal = True
event_butane_depleted.direction = -1


def event_oxygen_depleted(V, z):
    # Stop integration when oxygen molar flow reaches zero
    xi = z[0:3]
    ndot = ndot0 + xi @ stoic_matrix
    return float(ndot[1]) # O2

event_oxygen_depleted.terminal = True
event_oxygen_depleted.direction = -1

# set conditions at inlet (in base SI units)

P0 = 2.0

ndot0 = np.array([12.0, 137.48, 0, 0, 0, 0, 517.19])

# specify number of data points for displayed solution
ndata = 500

Ts = np.linspace(670, 673.5, 8) # K
Vs = np.linspace(1.9, 3, 12) # m^3

MA_grid = np.zeros((len(Ts), len(Vs)))
Y_grid = np.zeros_like(MA_grid)
S_grid = np.zeros_like(MA_grid)
XB_grid = np.zeros_like(MA_grid)

```

```
for i, T0 in enumerate(Ts):
    for j, Vreactor in enumerate(Vs):
        V_span = (0.0, float(Vreactor))
        z0 = [0, 0, 0, T0]
        sol2 = solve_ivp(
            rhs_adiabatic,
            V_span,
            z0,
            method="BDF",
            events=[event_butane_depleted, event_oxygen_depleted],
            rtol=1e-6,
            atol=1e-8,
        )

        if not sol2.success or sol2.y.size == 0:
            MA_grid[i, j] = np.nan
            Y_grid[i, j] = np.nan
            S_grid[i, j] = np.nan
            XB_grid[i, j] = np.nan
            continue

        # Outlet state = last successfully computed point (either Vreactor or event stop)
        xi = sol2.y[0:3, -1]
        T = sol2.y[-1, -1]

        MA_out = xi[0] - xi[1]
        B_cons = xi[0] + xi[2]

        if B_cons <= 0:
            S_MA = 0.0
            Y_MA = 0.0
            X_B = 0.0
        else:
            S_MA = MA_out / B_cons
            Y_MA = MA_out / ndot0[0]
            X_B = B_cons / ndot0[0]

        MA_grid[i, j] = MA_out
        Y_grid[i, j] = Y_MA
        S_grid[i, j] = S_MA
        XB_grid[i, j] = X_B

# combined objective: yield  selectivity
J_grid = Y_grid * S_grid

# contour plot
V_mesh, T_mesh = np.meshgrid(Vs, Ts)

# 1) Contour plot for conversion to maleic anhydride (yield on feed) (darker = higher)
plt.figure()
levels_y = np.linspace(0.0, np.nanmax(Y_grid), 120)
cf = plt.contourf(V_mesh, T_mesh, Y_grid, levels=levels_y, cmap="tab20", vmin=0.0, vmax=np.nanmax(Y_grid
    ))
plt.colorbar(cf, label="Yield␣to␣MA")
plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
plt.ylabel("Temperature␣T␣(K)")
plt.title("Conversion␣to␣maleic␣anhydride␣(1␣-␣max)")
plt.tight_layout()
plt.show()

# 2) MA production rate with butane conversion contour lines (for readability)
plt.figure()
levels_ma = 120
cf = plt.contourf(V_mesh, T_mesh, MA_grid, levels=levels_ma, cmap="tab20b")
plt.colorbar(cf, label="MA␣out␣(mol/s)")

plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
plt.ylabel("Temperature␣T␣(K)")
plt.title("MA␣out␣with␣butane")
```

```
plt.tight_layout()
plt.show()

# 3) MA yield with butane conversion contour lines (for readability)
plt.figure()
levels_ma = 120
cf = plt.contourf(V_mesh, T_mesh, XB_grid, levels=levels_ma, cmap="tab20c")
plt.colorbar(cf, label="Butane conversion (out of 1)")

plt.xlabel("Reactor volume V (m$^3$)")
plt.ylabel("Temperature T (K)")
plt.title("Butane conversion contours")
plt.tight_layout()
plt.show()

# 4) Contour plot of selectivity to maleic anhydride (yield on feed) (darker = higher)
plt.figure()
levels_y = np.linspace(0.0, np.nanmax(S_grid), 120)
cf = plt.contourf(V_mesh, T_mesh, S_grid, levels=levels_y, cmap="tab20", vmin=0.0, vmax=np.nanmax(Y_grid
    ))
plt.colorbar(cf, label="Selectivity to MA")
plt.xlabel("Reactor volume V (m$^3$)")
plt.ylabel("Temperature T (K)")
plt.title("Selectivity to MA with butane")
plt.tight_layout()
plt.show()

# 5) Contour plot of J-coefficient (darker = higher)
plt.figure()
levels_y = np.linspace(0.0, np.nanmax(J_grid), 120)
cf = plt.contourf(V_mesh, T_mesh, J_grid, levels=levels_y, cmap="tab20", vmin=0.0, vmax=np.nanmax(Y_grid
    ))
plt.colorbar(cf, label="Selectivity to MA")
plt.xlabel("Reactor volume V (m$^3$)")
plt.ylabel("Temperature T (K)")
plt.title("J-coefficient analysis contours")
plt.tight_layout()
plt.show()


# -------------------------
# Optimal operating curves (reduce 2D grid to 1D optimum slices)
# -------------------------

# 1) Best V for each T0
V_opt_per_T = np.full(len(Ts), np.nan)
Y_opt_per_T = np.full(len(Ts), np.nan)
for i in range(len(Ts)):
    row = J_grid[i, :]
    if np.all(np.isnan(row)):
        continue
    j_best = int(np.nanargmax(row))
    V_opt_per_T[i] = Vs[j_best]
    Y_opt_per_T[i] = row[j_best]

plt.figure()
sc = plt.scatter(Ts, V_opt_per_T, c=J_grid[np.arange(len(Ts)), np.nanargmax(J_grid, axis=1)], s=22)
plt.plot(Ts, V_opt_per_T, linewidth=1)
plt.colorbar(sc, label="J coefficient at optimum, J")
plt.xlabel("Inlet temperature T0 (K)")
plt.ylabel("Optimal reactor volume V* (m$^3$)")
plt.title("Optimal reactor volume V* as a function of inlet temperature T0")
plt.tight_layout()
plt.show()

# 2) Best T0 for each V
T_opt_per_V = np.full(len(Vs), np.nan)
Y_opt_per_V = np.full(len(Vs), np.nan)
for j in range(len(Vs)):
```

```
    col = J_grid[:, j]
    if np.all(np.isnan(col)):
        continue
    i_best = int(np.nanargmax(col))
    T_opt_per_V[j] = Ts[i_best]
    Y_opt_per_V[j] = col[i_best]

plt.figure()
sc = plt.scatter(Vs, T_opt_per_V, c=J_grid[np.nanargmax(J_grid, axis=0), np.arange(len(Vs))], s=22)
plt.plot(Vs, T_opt_per_V, linewidth=1)
plt.colorbar(sc, label="J coefficient at optimum, J")
plt.xlabel("Reactor volume V (m$^3$)")
plt.ylabel("Optimal inlet temperature T0* (K)")
plt.title("Optimal inlet temperature T0* as a function of reactor volume V")
plt.tight_layout()
plt.show()


# -------------------------
# Print all data for a specific grid point (V=37.5 m^3, T=668 K)
# -------------------------

V_target = 2.5
T_target = 671.5

# Find indices in the grids (robustly)
i_candidates = np.where(np.isclose(Ts, T_target, atol=1e-9))[0]
j_candidates = np.where(np.isclose(Vs, V_target, atol=1e-9))[0]

if len(i_candidates) == 0:
    raise ValueError(f"T_target={T_target} not found in Ts grid. Ts[0]={Ts[0]}, Ts[-1]={Ts[-1]}, step~{
        Ts[1]-Ts[0]}")
if len(j_candidates) == 0:
    raise ValueError(f"V_target={V_target} not found in Vs grid. Vs[0]={Vs[0]}, Vs[-1]={Vs[-1]}, step~{
        Vs[1]-Vs[0]}")

i = int(i_candidates[0])
j = int(j_candidates[0])

print("\n==============================")
print("Point data for (V=37.5 m^3, T=668 K)")
print("==============================")
print(f"Grid indices: i(T)={i}, j(V)={j}")
print(f"T_grid={Ts[i]:.6f} K, V_grid={Vs[j]:.6f} m^3\n")

print("From precomputed grids:")
print(f"MA_out (mol/s)           = {MA_grid[i, j]:.12g}")
print(f"Yield to MA (Y_MA)       = {Y_grid[i, j]:.12g}")
print(f"Selectivity to MA (S_MA) = {S_grid[i, j]:.12g}")
print(f"Butane conversion (X_B)  = {XB_grid[i, j]:.12g}")
print(f"J = Y*S                  = {J_grid[i, j]:.12g}")
```

A.3 Cooling system.py

```
# Lev Gerasimov's Project 3 code
# Right now I approximate the adiabatic with cooling situation of the reactor of maleic ahidryde
# February 2026

# 3 reactions in gas phase
# C4H10 + 3.5_O2 = C4H2O3 + 5_H2O
# C4H2O3 + 2_O2 = 2_CO + 2_CO2 + H2O
# C4H10 + 5_O2 = 3_CO + CO2 + 5_H2O
# list of species and their points below
# 0 - C4H10
# 1 - O2
# 2 - C4H2O3
# 3 - H2O
```

```
# 4 - CO
# 5 - CO2
# 6 - N2 (in air)


import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# number of species
nspec = 7

# number of reactions
nrec = 3

# stoichiometric coefficients converted into matrix
stoic_1 = np.array([-1, -3.5, 1, 4, 0, 0, 0])
stoic_2 = np.array([0, -2, -1, 1, 2, 2, 0])
stoic_3 = np.array([-1, -5, 0, 5, 3, 1, 0])

stoic_matrix = np.stack([stoic_1, stoic_2, stoic_3])

# Arrhenius coefficients and activation energy for three reactions
A_matrix = np.array([
    20.0, # Reaction 1: selective oxidation to maleic anhydride
    4.0, # Reaction 2: oxidation of maleic anhydride
    0.7 # Reaction 3: combustion of n-butane
])

E_act_matrix = np.array([
    95e3, # Reaction 1
    105e3, # Reaction 2
    125e3 # Reaction 3
])


def k_calc(A_matrix, E_act_matrix, T):
    T1 = T
    k_new = A_matrix * np.exp(-(E_act_matrix / 8.3145) * (1.0 / T1 - 1.0 / 673.15))
    return k_new


# rate coefficients for the reactions
KMA = 10
KB = 22
KS = 2

EPS = 1e-12

# Enthalpies of formation at 298 K (in base SI units)
delHf = np.array([
    -125.6e3, # n-Butane, kJ/mol -> J/mol
    0.0, # Oxygen
    -398.3e3, # Maleic anhydride (vapour)
    -241.8e3, # Water (steam)
    -110.5e3, # Carbon monoxide
    -393.5e3, # Carbon dioxide
    0.0 # Nitrogen
])

# Heat capacity coefficients (in base SI units)
# Cp_m(T) = a + b*T + c*T^2
# Each Cp_paramX is [a, b, c] for species X (see species numbering above).

Cp_param0 = np.array([3.79, 358e-3, -137e-6]) # 0 - n-Butane
Cp_param1 = np.array([26.5, 9.4e-3, -0.3e-6]) # 1 - O2
Cp_param2 = np.array([14.4, 294e-3, -138e-6]) # 2 - Maleic anhydride (vapour)
Cp_param3 = np.array([32.0, 3.2e-3, 6.6e-6]) # 3 - H2O (steam)
Cp_param4 = np.array([28.9, -1.2e-3, 6.3e-6]) # 4 - CO
```

```
Cp_param5 = np.array([22.3, 58.3e-3, -27.4e-6]) # 5 - CO2
Cp_param6 = np.array([29.4, -3.2e-3, 7.3e-6]) # 6 - N2

Cp_param_matrix = np.stack([
    Cp_param0,
    Cp_param1,
    Cp_param2,
    Cp_param3,
    Cp_param4,
    Cp_param5,
    Cp_param6
])


def Cp_calc(nspec, Cp_param_matrix, T):
    T_func = np.array([1, T, T**2])
    Cp_val = Cp_param_matrix @ T_func
    return Cp_val


def delHreac(stoic_matrix, delHf, cp_param_matrix, T):
    delHf298 = stoic_matrix @ delHf
    delCP = stoic_matrix @ cp_param_matrix
    T_func = np.array([T - 298.0, (T**2 - 298.0**2) / 2.0, (T**3 - 298.0**3) / 3.0])
    delH_reac = delHf298 + delCP @ T_func
    return delH_reac


# Cooling system parameters (simple overall heat transfer model)
U = 250.0 # W/(m^2 K) (example)
A_over_V = 80.0 # m^2/m^3 (example)
T_coolant = 653.0 # K (380C)


def rhs_cooling(V, z):
    P = P0
    xi = z[0:3]
    T = z[-1]

    ndot = ndot0 + xi @ stoic_matrix
    if np.any(ndot < -1e-6):
        return np.zeros(4)
    ndot = np.maximum(ndot, 0.0)

    ndot_tot = float(ndot.sum())
    if ndot_tot <= EPS:
        return np.zeros(4)

    y = ndot / ndot_tot
    y = np.maximum(y, 0.0)
    y_sum = float(y.sum())
    if y_sum <= EPS:
        return np.zeros(4)
    y = y / y_sum

    P_part = np.maximum(y * P, 0.0)

    PB = P_part[0]
    PMA = P_part[2]
    PS = P_part[3]

    k1, k2, k3 = k_calc(A_matrix, E_act_matrix, T)
    r1 = k1 * PB / (1 + KB * PB + KMA * PMA + KS * PS)
    r2 = k2 * PMA
    r3 = k3 * PB

    R_tot = np.array([r1, r2, r3])

    Cp = Cp_calc(nspec, Cp_param_matrix, T)
```

```
    Cpdot = float(ndot @ Cp)
    if Cpdot <= EPS or not np.isfinite(Cpdot):
        return np.zeros(4)

    Hreact = delHreac(stoic_matrix, delHf, Cp_param_matrix, T)

    # heat removal term (overall model)
    Qrem = U * A_over_V * (T - T_coolant) # W per m^3
    dT = (-float(np.dot(Hreact, R_tot)) - Qrem) / Cpdot
    if not np.isfinite(dT):
        return np.zeros(4)

    return np.array([r1, r2, r3, dT])


def event_butane_depleted(V, z):
    xi = z[0:3]
    ndot = ndot0 + xi @ stoic_matrix
    return float(ndot[0])


event_butane_depleted.terminal = True
event_butane_depleted.direction = -1


def event_oxygen_depleted(V, z):
    xi = z[0:3]
    ndot = ndot0 + xi @ stoic_matrix
    return float(ndot[1])


event_oxygen_depleted.terminal = True
event_oxygen_depleted.direction = -1


P0 = 2.0
ndot0 = np.array([12.0, 137.48, 0.0, 0.0, 0.0, 0.0, 517.19])

Ts = np.linspace(660, 700, 101)
Vs = np.linspace(0.1, 20, 101)

MA_grid = np.zeros((len(Ts), len(Vs)))
Y_grid = np.zeros_like(MA_grid)
S_grid = np.zeros_like(MA_grid)
XB_grid = np.zeros_like(MA_grid)

for i, T0 in enumerate(Ts):
    for j, Vreactor in enumerate(Vs):
        V_span = (0.0, float(Vreactor))
        z0 = [0.0, 0.0, 0.0, float(T0)]
        sol = solve_ivp(
            rhs_cooling,
            V_span,
            z0,
            method="BDF",
            events=[event_butane_depleted, event_oxygen_depleted],
            rtol=1e-6,
            atol=1e-8,
        )

        if not sol.success or sol.y.size == 0:
            MA_grid[i, j] = np.nan
            Y_grid[i, j] = np.nan
            S_grid[i, j] = np.nan
            XB_grid[i, j] = np.nan
            continue

        xi_end = sol.y[0:3, -1]
        MA_out = float(xi_end[0] - xi_end[1])
```

```
        B_cons = float(xi_end[0] + xi_end[2])

        if B_cons <= 0:
            S_MA = 0.0
            Y_MA = 0.0
            X_B = 0.0
        else:
            S_MA = MA_out / B_cons
            Y_MA = MA_out / float(ndot0[0])
            X_B = B_cons / float(ndot0[0])

        MA_grid[i, j] = MA_out
        Y_grid[i, j] = Y_MA
        S_grid[i, j] = S_MA
        XB_grid[i, j] = X_B

J_grid = Y_grid * S_grid

V_mesh, T_mesh = np.meshgrid(Vs, Ts)

plt.figure()
cf = plt.contourf(V_mesh, T_mesh, Y_grid, levels=120, cmap="tab20")
plt.colorbar(cf, label="Yield␣to␣MA")
plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
plt.ylabel("Temperature␣T␣(K)")
plt.title("Cooling␣case:␣Yield␣to␣MA")
plt.tight_layout()
plt.show()

plt.figure()
cf = plt.contourf(V_mesh, T_mesh, MA_grid, levels=120, cmap="tab20b")
plt.colorbar(cf, label="MA␣out␣(mol/s)")
plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
plt.ylabel("Temperature␣T␣(K)")
plt.title("Cooling␣case:␣MA␣out")
plt.tight_layout()
plt.show()

plt.figure()
cf = plt.contourf(V_mesh, T_mesh, XB_grid, levels=120, cmap="tab20c")
plt.colorbar(cf, label="Butane␣conversion")
plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
plt.ylabel("Temperature␣T␣(K)")
plt.title("Cooling␣case:␣Butane␣conversion")
plt.tight_layout()
plt.show()

plt.figure()
cf = plt.contourf(V_mesh, T_mesh, S_grid, levels=120, cmap="tab20")
plt.colorbar(cf, label="Selectivity␣to␣MA")
plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
plt.ylabel("Temperature␣T␣(K)")
plt.title("Cooling␣case:␣Selectivity␣to␣MA")
plt.tight_layout()
plt.show()

plt.figure()
cf = plt.contourf(V_mesh, T_mesh, J_grid, levels=120, cmap="tab20")
plt.colorbar(cf, label="J␣coefficient")
plt.xlabel("Reactor␣volume␣V␣(m$^3$)")
plt.ylabel("Temperature␣T␣(K)")
plt.title("Cooling␣case:␣J␣coefficient")
plt.tight_layout()
plt.show()
```