Conceptual Design and Techno-economic Assessment of Benzene Production by Hydrodealkylation (HDA) of Toluene Using an Adiabatic Plug Flow Reactor

Group 14

Report Submitted on April 30, 2025

Author:	**Position:**
Sean Shen	Process Development Engineer
Yulun Wu	Process Development Engineer
Chang Yuan	Process Development Engineer

---

1 Exectutive Summary

This report presents a comprehensive techno-economic assessment of a benzene production facility utilizing hydrodealkylation (HDA) of toluene with an adiabatic plug flow reactor. Alkyl Products Limited (APL) aims to expand its benzene production capacity to 100,000 metric tonnes per year, evaluating process feasibility, capital investment, and profitability. The HDA process converts toluene and hydrogen at high temperatures and pressures into benzene and methane. This design employs an adiabatic reactor to optimize capital costs by eliminating external heat exchange. A gas recycle and purge system is implemented to reduce raw material consumption. Heat integration further minimizes energy demand, and diphenyl byproduct is burned as fuel to enhance sustainability.

The estimated total capital investment (TCI) is \$31.2 million, consisting of a fixed capital cost of \$19.6 million, working capital of \$9.6 million, and start-up costs of \$2.0 million. The discounted cash flow (DCF) analysis projects a net present value (NPV) of \$49.9 million over a 15-year project life, with an internal rate of return (IRR) of 40%. The expected payback period is between four and five years. Key operating costs include raw materials, such as toluene at \$300 per tonne and hydrogen feed at \$1,400 per tonne (95% H_2/5% CH_4 mixture). Benzene is expected to sell at \$1,000 per tonne, maintaining favorable margins. A \$40 per tonne CO_2 emissions charge is incorporated for sustainability compliance. The energy consumption for the production of benzene is estimated at 7 MJ per kilogram, largely attributed to furnace heating and hydrogen compression. CO_2 emissions are calculated at 0.4–0.5 kg per kg of benzene, considering both direct combustion and indirect electricity usage. The inclusion of waste fuel utilization and energy recovery systems significantly enhances environmental performance.

Contents

1 Exectutive Summary **1**

2 Introduction **4**
2.1 Motivation and Background 4
2.2 Market Analysis 4
2.3 Reaction Chemistry 5

3 Conceptual design **6**
3.1 Process Overview 6
3.2 Conceptual Design for Reactor and Recycle System 7
3.3 Conceptual Design of Distillation Column 9
3.4 Conceptual Design for Heat Exchangers 10
3.5 Detailed Flowsheet Generated by HYSYS 11
3.6 Comparison of Conceptual Design with HYSYS Simulation 14

4 Economy Analysis **15**
4.1 Capital and Operation Cost Analysis 15
4.2 Cash Flow and Internal Rate of Return Analysis 16
4.3 Sensitivity Analysis 17

5 Process Safety and Harzards **18**
5.1 Chemical Hazards and Environmental Impact 19

6 Process Alternatives and Next Steps **19**

7 Conclusions **20**

8 Appendix **22**
8.1 Appendix A: Detailed Mole Balance for HDA Process 22
8.1.1 Reactor Model Setup 22
8.1.2 Basis Simulation (1 mol/s Toluene) 22
8.1.3 Selectivity and Conversion 22
8.1.4 Scaling to Desired Production Rate 23
8.1.5 True Reactor Outlet Profiles 23
8.1.6 Plant-Level Mass Balances: Recycle and Purge 23
8.1.7 Outlet Stream Destination 24
8.2 Appendix B: Reaction Kinetics and Thermodynamics 24
8.2.1 Reaction Kinetics[**2**] 24
8.2.2 Reaction Thermodynamics 26
8.3 Appendix C: Estimation of the Underwood binary distillation[**14**] 27
8.3.1 Calculation of Relative Volatility 27
8.3.2 Reflux Ratio and Boil-Up Ratio Calculation 28
8.3.3 Calculation of Rectifying and Stripping Stages 29
8.3.4 Vapor Flowrates and Energy Balances 30
8.3.5 Cross-Sectional Area Calculation 32
8.3.6 Column Diameter and Height Estimation 33

8.3.7 Calculation of the Heating Exchange Area for Condenser and Reboiler . . . 34
8.4 Appendix D: Capital Cost of Process Equipment[**12**] . . . 34
8.4.1 Process Furnaces . . . 34
8.4.2 Direct-Fired Heaters . . . 35
8.4.3 Heat Exchangers . . . 36
8.4.4 Gas Compressors . . . 36
8.4.5 Pressure Vessels, Columns, and Reactors . . . 37
8.4.6 Separation Cost . . . 37
8.4.7 Distillation Column Trays and Tower Internals . . . 37
8.5 Appendix E: Economic Assumption, Formular and Spreadsheet . . . 39
8.5.1 Economic Set UP . . . 39
8.5.2 Product Sales Revenue . . . 40
8.5.3 Feedstock Cost . . . 41
8.5.4 Profit Calculation . . . 41
8.5.5 Equipment Cost Estimation . . . 41
8.5.6 Annual Depreciation . . . 41
8.6 Appendix H: Detailed Cash Flow Sheet . . . 44
8.7 Appendix I: Python Code . . . 45
8.7.1 Mole balance and Reactor Design . . . 45
8.7.2 Ternary Distillation System USing Underwood . . . 58
8.7.3 Binary Distillation Using Underwood Equation . . . 66
8.7.4 Heat Exchanger Optimization . . . 73

2 Introduction

2.1 Motivation and Background

Hydrodealkylation (HDA) of toluene is a more cost-effective method to produce benzene, particularly compared to catalytic reforming, one of the major methods to produce benzene in industry. Unlike catalytic reforming, which produces a mixture of aromatics and requires complex separation processes, HDA offers high selectivity toward benzene with minimal formation of heavy aromatics or other byproducts[**1**]. Furthermore, HDA utilizes relatively inexpensive and widely available feedstocks, toluene and hydrogen, and can be more easily adjusted to meet the fluctuating demand for benzene, making it a favorable choice for dedicated benzene production.

This design project aims to develop a conceptual process for the production of 100,000 metric tonnes per year of benzene using the HDA reaction at Alkyl Products Limited's (APL) Gulf Coast site in Freeport, TX. The process will employ an adiabatic reactor and a gas recycling and purge system will be implemented without a pressure swing adsorption (PSA) unit. The project follows a Front-End Loading Phase 1 (FEL-1) approach, providing a techno-economic evaluation to guide investment decisions. The primary objectives of this study include evaluating the feasibility and profitability of the proposed benzene production process, optimizing reaction conditions to maximize yield, and assessing environmental and safety considerations. Sustainability factors, such as carbon emissions, will also be analyzed, aligning with the commitment of APL to reduce its environmental footprint. The results of this study will inform future phases of development and serve as a foundation for scaling the process to full industrial implementation.

2.2 Market Analysis

Benzene is a highly demanded chemical feedstock, essential for the production of a wide range of industrial and consumer products. The global benzene market is primarily driven by its applications in producing derivatives such as styrene, cumene, cyclohexane, and nitrobenzene, which are used in the manufacturing of plastics, resins, synthetic fibers, adhesives, and pharmaceuticals[**2**].

According to recent industry reports, the global benzene market is expected to grow steadily as a result of the rising demand from the petrochemical and polymer industries. The increase in plastic consumption, particularly in the packaging, construction and automotive sectors, has significantly influenced the demand for benzene. The price of benzene is subject to fluctuations based on crude oil and naphtha prices, as well as supply chain dynamics. In recent years, supply constraints caused by refinery shutdowns and geopolitical tensions have affected the availability of benzene, leading to price volatility[**3**].

In addition, environmental regulations are shaping market trends, with stricter emissions standards and sustainability initiatives driving the need for greener production technologies. Thus, when we consider the economic profits of the HDA process, carbon dioxide emission has to be included. The market price of key chemicals used in the process is summarized below.

Table 1: HDA Substance Unit Prices

Substance	**Price**
Benzene (Selling Price)	\$1,000/tonne
Toluene (Feedstock Cost)	\$300/tonne
Hydrogen Gas Feed (95% H_2, 5% CH_4)	\$1,400/tonne of mixture
CO_2 Charge	\$40/tonne
Fuel	\$4.25/GJ

2.3 Reaction Chemistry

The benzene production process utilizes the hydrodealkylation (HDA) of toluene, a high-temperature, high-pressure gas-phase reaction in which toluene reacts with hydrogen to form benzene and methane:

$$\text{Toluene} + \text{H}_2 \rightarrow \text{Benzene} + \text{CH}_4 \quad \Delta H^\circ_{298} = -41.76 \text{ kJ/mol} \tag{1}$$

However, an undesirable reversible side reaction also occurs, in which benzene forms diphenyl and hydrogen:

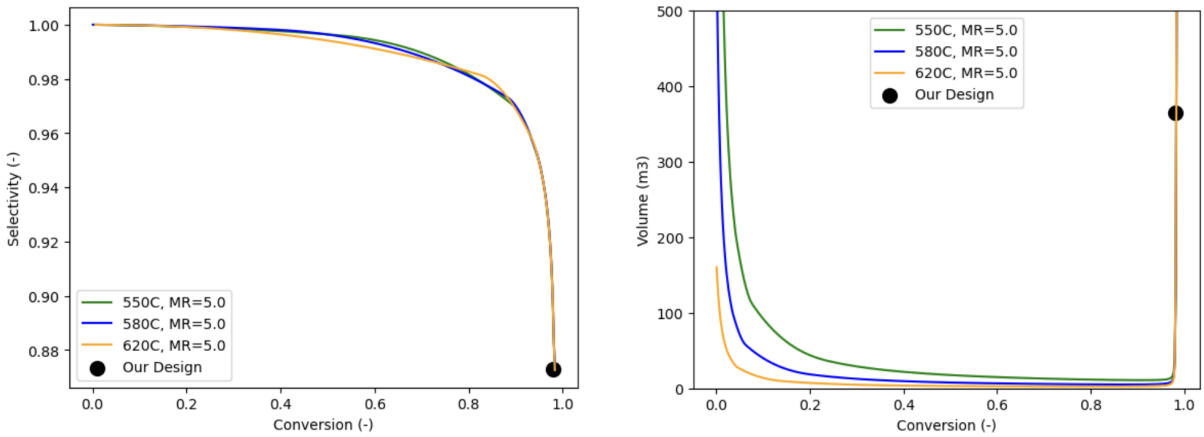
$$2 \text{ Benzene} \rightleftharpoons \text{Diphenyl} + \text{H}_2 \quad \Delta H^\circ_{298} = +12.63 \text{ kJ/mol} \tag{2}$$

The main HDA reaction is strongly exothermic, which means that as the reaction proceeds in the adiabatic plug flow reactor, the temperature rises significantly. This temperature increase accelerates the reaction kinetics via the Arrhenius relationship, thus increasing the rate constant k for both desired and undesired reactions. Specifically, higher temperatures would favor the formation of diphenyl due to the endothermic nature of the side reaction.

To balance these effects, the reactor is designed to operate within a temperature range of 550–650°C and a pressure range of 30–34 bar. An excess of hydrogen (typically a 5:1 hydrogen:toluene molar ratio) is maintained to shift both reactions toward the products. Operation above 650°C may lead to wall fouling and potential coking, which can cause performance degradation and increased maintenance. Therefore, the temperature rise resulting from the exothermic reaction must be carefully controlled to remain below this upper limit when designing the PFR.

Thermodynamic and kinetic parameters—including equilibrium constants and rate constants as functions of temperature—have been evaluated and are provided in Appendix B. These calculations guide the reactor optimization, ensuring that the desired benzene production target of 100,000 tonnes per annum is met while preventing excessive temperature rise and minimizing side-product formation.

After simulating the reaction kinetics in python, the relationship between conversion and selectivity is shown in Figure 1 (the reaction kinetics are listed in the Appendix B).



(a) Selectivity vs. Conversion **(b)** Conversion vs. Reactor Volume

Figure 1: Comparison of a) plant selectivity vs. conversion and b) conversion vs. reactor volume under different temperatures. Design parameters are obtained around point optimum.

The relationship between selectivity and reactor conversion in the adiabatic HDA process reveals a fundamental trade-off. At low conversion levels, selectivity toward benzene is high, indicating minimal side reactions and efficient toluene utilization. However, as conversion increases beyond 40%, selectivity begins to decline due to the onset of secondary reactions that generate unwanted by-products. A sharp drop in selectivity is observed as conversion approaches completeness, primarily due to the increasing concentration of reactive intermediates and thermodynamic limitations. At higher reactor temperatures, selectivity improves as a result of increased rate constants (k) for the desired reaction, which kinetically favors benzene formation over side reactions. This trend highlights the importance of operating the reactor at an optimal conversion level that balances high benzene yield with acceptable selectivity. Implementing recycle loop enhances hydrogen utilization and limits the extent of conversion per pass, thereby suppressing secondary reactions. A well-optimized adiabatic reactor, when combined with effective separation and recycle strategies, ensures high product purity, minimizes by-product formation, and enhances both the efficiency and profitability of the hydrodealkylation (HDA) process. One of the most critical aspects in reactor design is the choice of reactor volume and conversion. The observed trend indicates that at lower conversion levels (0–0.6), the required reactor volume increases gradually. However, as conversion approaches unity, the reactor volume rises exponentially. In our case, from a design standpoint, operating the reactor at around 95% conversion, strikes an optimal balance between conversion efficiency and reactor size, and helps to control both capital and operational costs. Additionally, reactor temperature must be carefully managed, as excessive temperatures can lead to reactor degradation and increased maintenance, compromising long-term performance.

3 Conceptual design

3.1 Process Overview

Figure 2 shows a simplified version of the HDA process.

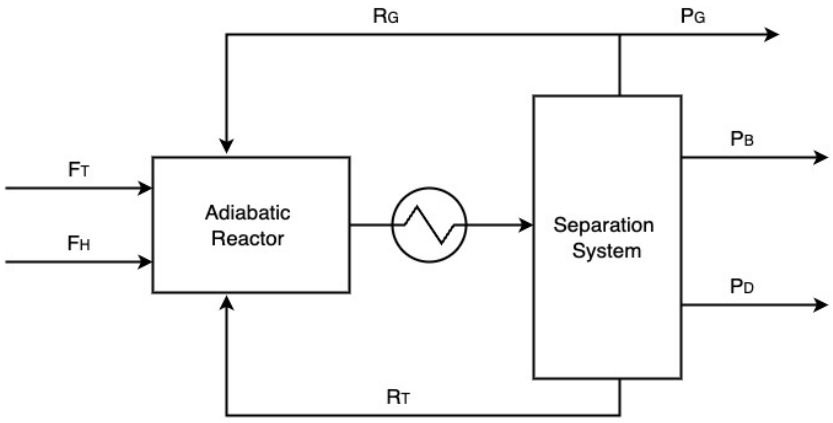


Figure 2: simple Process flow diagram for the HDA of Toluene

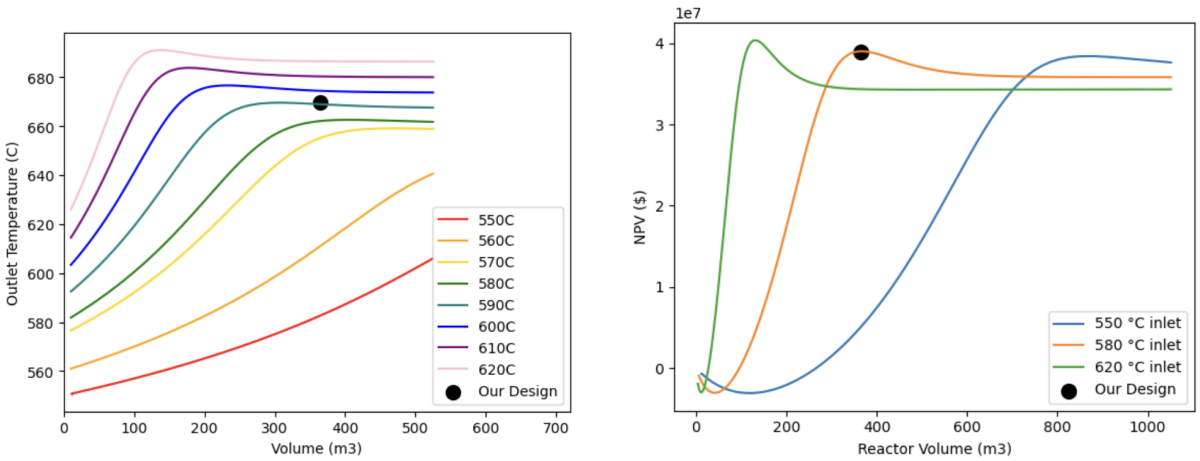
The process design consists of an adiabatic reactor and a separation system for the HDA of toluene, optimized for maximum benzene yield while efficiently managing byproducts and recycle streams. The fresh feed consists of pure toluene (135 kta) and a hydrogen-rich gas mixture (23.8 kta, 95 mol% H_2, 5 mol% CH_4), introduced at the reactor inlet to achieve a target benzene production of 100 kta. Within the adiabatic reactor, the exothermic HDA reaction takes place, converting toluene and hydrogen into benzene and methane. Due to side reactions, a small fraction of benzene form diphenyl as an undesirable byproduct. Since the reactor operates without external heat exchange, reaction heat sustains the required temperature profile.

The reactor effluent, consisting of unreacted hydrogen, methane, benzene, toluene, and diphenyl, is cooled by exchanging heat with the cold streams before entering the separation system. The separation unit isolates high-purity benzene as the primary product stream, while unreacted toluene and hydrogen are recycled back to the reactor to enhance raw material utilization. The recycled gas stream accumulates methane, necessitating a controlled purge to maintain mass balance. To minimize waste, diphenyl is separated and burned as fuel.

3.2 Conceptual Design for Reactor and Recycle System

The reactor selected for this process is a plug flow reactor (PFR), which offers higher conversion efficiency compared to a continuously stirred tank reactor (CSTR) of the same reactor volume. This choice is particularly advantageous for achieving the target production of 100 kilotonnes per annum (kta) of benzene. The reactor is designed to operate adiabatically, as specified by the project scope, and is paired with a recycle-purge system rather than a pressure swing adsorption (PSA) unit.

There are both advantages and drawbacks to this configuration. A major advantage of the adiabatic PFR is that the heat released from the highly exothermic primary reaction



(a) Outlet temperature vs. volume at varying inlet Temperature.

(b) Net present value (NPV) vs. volume at varying inlet Temperature.

Figure 3: Key performance metrics plotted against reactor conversion and volume for different inlet-temperature cases. (a) Outlet temperature vs. volume. (b) NPV vs. volume. The black dots indicate the operating condition in the base case MATLAB design.

can be utilized to accelerate reaction rates and reduce the external heating duty. This is especially beneficial given that the secondary reaction is endothermic and less favorable. However, the absence of a PSA system means methane cannot be fully removed from the system, allowing it to accumulate in the recycle loop. This buildup reduces the thermodynamic driving force for benzene formation, thereby limiting conversion. While our current design adheres to the assigned constraints, we recognize that a hybrid system combining an adiabatic reactor with PSA could offer improved performance by selectively removing methane.

To shift the equilibrium toward benzene production, an excess of hydrogen is required. We adopted a hydrogen-to-toluene molar ratio (MR) of 5. The operating pressure was set at 34 bar, in alignment with the available gas feed conditions, thus avoiding the need for an upstream compressor. The reactor inlet temperature was selected as 580°C, balancing several factors: trying to limiting the outlet temperature below 650°C to avoid thermal degradation (Figure 3a), ensuring sufficient reaction rates to meet production targets, and optimizing equipment and operating costs to maximize net present value (Figure 3b). Sensitivity analyses were performed across various operating conditions, and the selected specifications provided the most economically viable solution while satisfying all process constraints.

Under the base-case conditions, we observe that the reactor outlet temperature exceeds the specified 650°C limit by approximately 20°C. While the ideal solution would be to lower the inlet temperature to stay within this limit, doing so would either require an impractically large reactor or compromise our production goal of 100 kta of benzene. Given this tradeoff, we accept the modest temperature exceedance, as the primary concern is coke formation rather than equipment failure. Moreover, since the plant is scheduled to operate for 8,000 hours per year—leaving regular downtime for decoking—the risk is considered manageable at this stage. Nevertheless, operating above the design temperature introduces potential safety and fouling concerns, which should be thoroughly evaluated in subsequent design phases.

The hot reactor effluent is split to preheat both the gas and toluene feed streams, comprising fresh feed and recycled components, via heat exchangers. Supplementary furnaces are included to raise the final inlet streams to the desired 580°C before entering the reactor. This integrated heat recovery approach significantly reduces energy consumption and operating costs.

A critical aspect of the recycle system design is the calculation of the purge fraction. Since methane is introduced through the fresh gas feed and additionally formed in the reactor, the purge stream must be sized to balance this input and prevent methane accumulation in the recycle loop. We calculated the purge ratio dynamically, based on the methane content in both the outlet and fresh feed, ensuring the system reaches a steady-state without excessive loss of valuable hydrogen. Recycled gas and toluene streams are compressed to 34 bar before mixing with the fresh feeds, which are delivered at the same pressure, completing the feed preparation for the reactor.

3.3 Conceptual Design of Distillation Column

To design a profitable distillation system, Separation order of the components need to be decided. There are five components presented in the outlet feed: hydrogen, methane, toluene, benzene, and diphenyl. Hydrogen and methane, being non-condensable gases under ambient conditions, were separated using a gas-liquid flash drum. However, flash separation is ineffective for the remaining hydrocarbons, necessitating the use of distillation columns.

The relative volatilities of toluene, benzene, and diphenyl were calculated using the Antoine equation and Raoult's law, as detailed in Appendix C. Diphenyl, being the heaviest component, was chosen as the reference species. The relative volatilities of toluene and benzene with respect to diphenyl were found to be 27.2 and 53.7, respectively. A direct separation scheme was selected, where benzene is recovered as the distillate while toluene and diphenyl are withdrawn as bottom products. This decision is justified by the difficulty in separating toluene and benzene due to their similar volatilities and the much larger quantity of benzene relative to toluene in the feed. In contrast, separating toluene from diphenyl in a binary system is more feasible due to their greater volatility difference and relative abundances.

The conceptual design of the distillation column is based on the Underwood equations (Appendix C). The reflux ratio is set equal to twice the minimum reflux ratio of to ensure effective separation. The number of theoretical stages required was estimated by solving fulk set underwood equation. Using the O'Connell correlation, the overall column efficiency E_0 was approximated as 0.3, leading to an estimated 17 actual stages.

The vapor flowrate derived from the Underwood method was then used to size the column cross-sectional area and the required heat exchange areas for the condenser and reboiler. The overall heat transfer coefficients were taken from Table 9: 770 W/m^{2}K for the condenser and 820 W/m^{2}K for the reboiler. Figure 3(a) and (b) shows the total installed cost of the heat exchanger system as a function of the average temperature difference (ΔT_{avg}) and installed cost for distillation column based on tray materials and sizing.

Given the corrosive nature of toluene, benzene, and diphenyl, stainless steel was selected as the column material for its balance of corrosion resistance and cost. Sieve trays

were used to minimize capital investment while maintaining operability. For the heat exchanger(condenser and reboiler), a carbon steel shell and Monel tubes were selected, as this configuration offers a lower installed cost compared to fully stainless steel construction. This strategy was also applied to the binary distillation column for separating toluene and diphenyl.

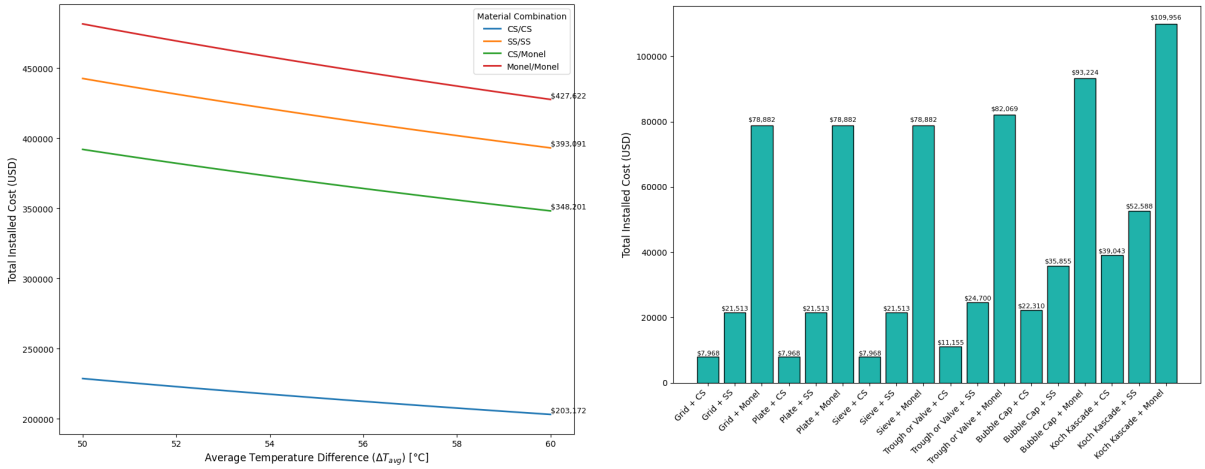


Figure 4: Cost breakdowns of ternary distillation column. (a) Heat exchanger material comparison. (b) Tray type and material combination analysis.

3.4 Conceptual Design for Heat Exchangers

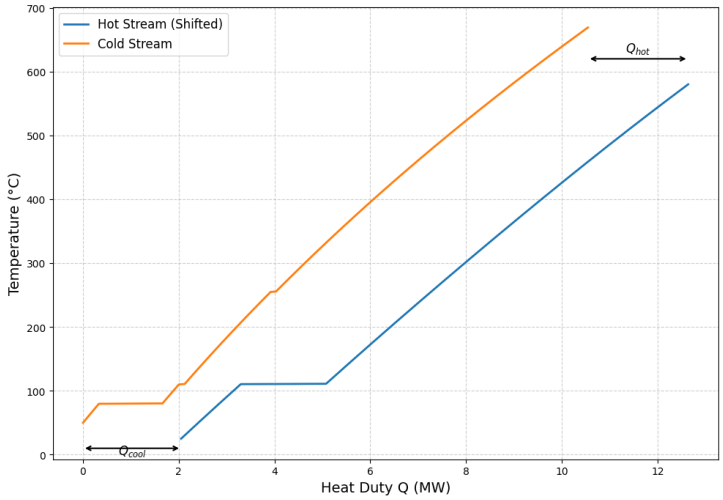


Figure 5: Composite Curves for Feed-Product Heat Integration with Pinch Point Indicated

To improve the thermal efficiency of the process, a three-unit heat exchanger network was implemented around the single adiabatic reactor. This integration strategy dramatically reduced utility consumption. Based on simulation results, the heating duty was reduced by 76.3% and the cooling duty by 97.5% relative to the no heat exchanger implementation. The design was guided by a pinch analysis performed in python, with the corresponding composite curve shown in Figure 5.

This process benefits from the adiabatic nature of reactor. The effluent exits at a temperature at 669 C, significantly higher than the temperature of the feed entering the reactor at 580 C, providing an opportunity for substantial heat recovery.

Two of the heat exchangers utilize this hot reactor effluent to preheat the incoming cold feed. To avoid pre-reacting, the two feed are heated separately. This direct integration between the feed and product streams significantly reduce the heat duty for external heating utilities. The third exchanger is placed after the reactor to cool down the hot product stream using cooling water at 35C, a necessary step before the product enters seperation system and recycled back as part of the feed.

The materials of construction were selected based on corrosion resistance and cost considerations. The two exchangers involved in feed-effluent integration were specified as stainless steal on both side, representing a cost-effective substitute for more exotic materials like tantalum. The cooling exchanger was designed as a Monel and carbon steel as tube and shell system with water as the coolant since Monel and carbon steel system is cheaper than using both stainless steel.

3.5 Detailed Flowsheet Generated by HYSYS

Table 1 shows the detailed flow table,Figure 7 shows process flow diagram generated by HYSYS, and Table 3 shows the Equipment Specifications:

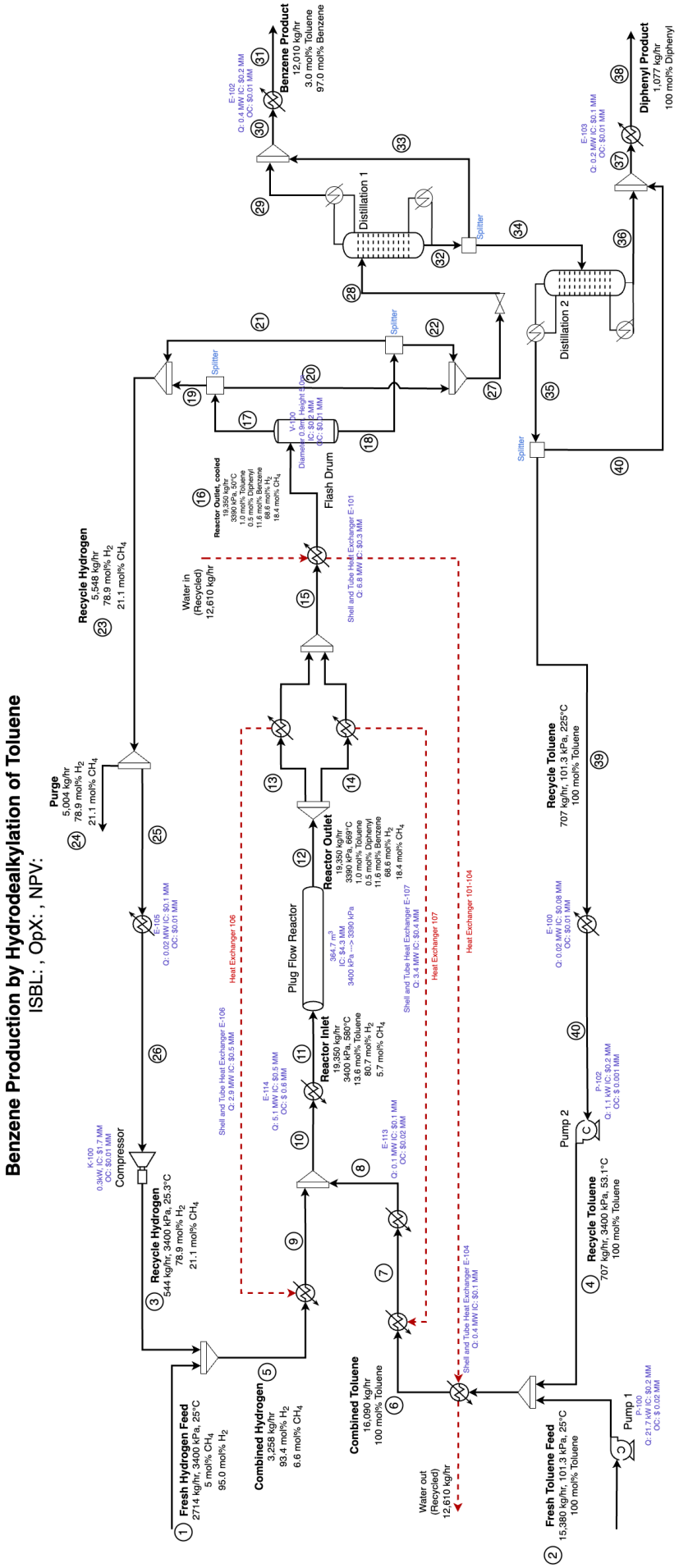


Figure 6: Process flow diagram for the HDA process, detailing the key equipment and streams involved in benzene production.

Table 2: Streams for HYSYS Simulation of the HDA Process

#	Stream Name	Temp (°C)	Pressure (kPa)	Stream Composition	Mass Flow (kg/hr)	Mass Flow (MT/year)	Molar Flow (kgmol/hr)
1	Hydrogen feed	25	3400	95.0 mol% H_2 5.0 mol% CH_4	2714	21712	998.9
2	Toluene feed	25	101.3	100.0 mol% Toluene	15380	123040	167
3	Recycle hydrogen	25.32	3400	78.9 mol% H_2 21.1 mol% CH_4	543.7	4349.6	109.2
4	Recycle toluene	53.07	3400	100.0 mol% Toluene	707	5656	7.673
5	Combined hydrogen inlet	25.03	3400	93.4 mol% H_2 6.6 mol% CH_4	3258	26064	1108
6	Combined toluene inlet	80.09	3400	100.0 mol% Toluene	16090	128720	174.6
7	Toluene inlet heated 1	90	3400	100.0 mol% Toluene	16090	128720	174.6
8	Toluene inlet heated 2	309.5	3400	100.0 mol% Toluene	16090	128720	174.6
9	Hydrogen inlet heated 1	351.2	3400	93.4 mol% H_2 6.6 mol% CH_4	3258	26064	1108
10	Reactor inlet (before trim)	330.5	3400	13.6 mol% Toluene 80.7 mol% H_2 5.7 mol% CH_4	19350	154800	1283
11	Reactor inlet	580	3400	13.6 mol% Toluene 80.7 mol% H_2 5.7 mol% CH_4	19350	154800	1283
12	Reactor outlet	669.9	3390	1.0 mol% Toluene 68.6 mol% H_2 11.6% mol Benzene 18.4% mol CH4 0.5%mol Diphenyl	19350	154800	1283
13	Split outlet 1	669.9	3390	1.0 mol% Toluene 68.6 mol% H_2 11.6% mol Benzene 18.4% mol CH4 0.5%mol Diphenyl	7740	61920	513.1
14	Split outlet 2	669.9	3390	1.0 mol% Toluene 68.6 mol% H_2 11.6% mol Benzene 18.4% mol CH4 0.5%mol Diphenyl	11610	92880	769.7
15	Reactor outlet (exchanged)	336.4	3390	1.0 mol% Toluene 68.6 mol% H_2 11.6% mol Benzene 18.4% mol CH4 0.5%mol Diphenyl	19350	154800	1283
16	Reactor outlet (cooled)	50	3390	1.0 mol% Toluene 68.6 mol% H_2 11.6% mol Benzene 18.4% mol CH4 0.5%mol Diphenyl	19350	154800	1283
17	Flash top	50	3390	0.02 mol% Toluene 78.2 mol% H_2 1.0 mol% Benzene 20.8% CH4	6420	51360	1123
18	Flash bottom	50	3390	7.5 mol% Toluene 0.6 mol% H_2 86.0 mol% Benzene 1.5 mole%	12930	103440	160
19	RH1	50	3390	79.0 mol% H_2 21.0 mol% CH_4	5511	44088	1112
20	TDB1	50	3390	1.8 mol% Toluene 98.2 mol% Benzene	909.2	7273.6	11.6
21	RH2	50	3390	27.4 mol% H_2 72.6 mol% CH_4	41.9	335.2	3.375
22	TDB2	50	3390	7.7 mol% Toluene 87.9 mol% Benzene 4.4 mol% Diphenyl	12890	103120	14.45
23	Before purge	50	3390	78.9 mol% H_2 21.1 mol% CH_4	5548	44384	1114
24	Purge	50	3390	78.9 mol% H_2 21.1 mol% CH_4	5004.3	40034.4	1005
25	Warm recycle hydrogen	50	3390	78.9 mol% H_2 21.1 mol% CH_4	543.7	4349.6	109.2
26	Cooled recycle hydrogen	25	3390	78.9 mol% H_2 21.1 mol% CH_4	543.7	4349.6	109.2
27	TDB	50	3390	7.2 mol% Toluene 88.6 mol% Benzene 4.2 mol% Diphenyl	13800	110400	167.6
28	TDB decompressed	50	101.3	7.2 mol% Toluene 88.6 mol% Benzene 4.2 mol% Diphenyl	13800	110400	167.6
29	Distillate 1	79.28	101.3	3.0 mol% Toluene 97.0 mol% Benzene	11580	92640	147.5
30	Combined benzene	79.37	101.3	3.0 mol% Toluene 97.0 mol% Benzene	12010	96080	153
31	Benzene product	25	101.3	3.0 mol% Toluene 97.0 mol% Benzene	12010	96080	153
32	Bottom 1	110	101.3	38.1 mol% Toluene 27.2 mol% Benzene 34.7 mol% Diphenyl	2213	17704	20.15
33	Benzene remnant (Bottom 1)	245.3	101.3	100.0 mol% Benzene	427.9	3423.2	5.479
34	TD	48.18	101.3	52.4 mol% Toluene 47.6 mol% Diphenyl	1785	14280	14.67
35	Distillate 2	115.5	101.3	87.3 mol% Toluene 12.7 mol% Diphenyl	880.9	7047.2	8.804
36	Bottom 2	225.2	101.3	100.0 mol% Diphenyl	904.2	7233.6	5.864
37	Diphenyl, hot	236.3	101.3	100.0 mol% Diphenyl	1077	8616	6.987
38	Diphenyl product	30	101.3	100.0 mol% Diphenyl	1077	8616	6.987
39	RT1	110.6	101.3	100.0 mol% Toluene	707.8	5662.4	7.681
40	RT2	50	101.3	100.0 mol% Toluene	707.8	5662.4	7.681

Table 3: Equipment Specifications

Equipment	HYSYS Design	Installed Cost ($)	Operating Cost (MM$/yr)	Material
Distillation Column 1	Pressure: 1 bar Stages: 17 Condenser: 1.26 MW Reboiler: 1.35 MW V/F: 0.91	370,000	0.26	Stainless Steel
Distillation Column 2	Pressure: 1 bar Stages: 28 Condenser: 0.091 MW Reboiler: 0.155 MW V/F: 0.679	80,689	0.04	Stainless Steel
PFR-100	Reactor volume: 364.7 m^3	4,327,601	–	SS
E-100	0.02 MW	80,800	0.3	SS
E-101	60.32 m^2 heat exchange area, 6.8 MW	251,600	–	CS/Monel
E-102	0.4 MW	160,200	0.01	SS
E-103	0.2 MW	101,600	0.01	SS
E-104	60.32 m^2 heat exchange area, 0.4 MW	135,700	0.1	CS/Monel
E-105	0.02 MW	122,900	0.01	SS
E-106	60.32 m^2 heat exchange area, 2.9 MW	513,700	0.02	CS/Monel
E-107	52.78 m^2 heat exchange area, 3.4 MW	434,700	–	CS/Monel
E-113	0.1 MW	100,000	0.02	SS
E-114	5.1 MW	502,000	0.6	SS
P-100	21.7481 kW	193,200	0.01	SS
P-102	1.05504 kW	166,200	0.001	SS
V-100	Diameter: 0.9244 m Height: 5.029 m	200,100	0.01	CS/Monel
K-100	0.295421 kW	1,672,400	–	SS

3.6 Comparison of Conceptual Design with HYSYS Simulation

Below is the comparison table between the molar flow generated by HYSYS versus the molar flow generated by Matlab that shows comparison:

Table 4: Comparison of Molar Flow Rates and Percentage Differences

Stream	Molar Flow (kgmol/h) - matlab	Molar Flow (kgmol/h) - HYSYS	Percent Difference (%)
Toluene Feed	167	167	0
Hydrogen Feed	998.9	998.9	0
Reactor In	1202.2	1140.0	5.46
Reactor Out	1202.2	1140.0	5.46
Recycled Stream	101.4	116.7	13.11
Purge	925.2	1005	7.94
Toluene Recycle	1.3	7.7	492.31
Crude Benzene	151.4	153	1.05
Diphenyl	11.9	7.0	70

The discrepancies between the molar flowrates predicted by MATLAB and those generated by HYSYS can primarily be attributed to the thermodynamic models employed in each platform. In our MATLAB-based simulation, the ideal gas law was used as the equation of state (EOS), which assumes no intermolecular interactions and constant specific heats—suitable for high-temperature, low-pressure gas systems but less accurate under real process conditions. In contrast, HYSYS utilizes the more sophisticated Uniquac equation of state, which accounts for non-ideal phase behavior and provides a more rigorous representation of thermophysical properties, especially for complex multicomponent systems.

As a result of these fundamental modeling differences, the reactor conversion calculated in MATLAB was 0.983, whereas HYSYS reported a slightly lower conversion of 0.93.

This variation in conversion cascades through the material balance, explaining the modest differences observed in reactor inlet and outlet molar flows, as well as the purge and recycle streams. Notably, larger percent differences are seen in minor components such as toluene recycle and diphenyl, likely due to their relatively small flowrates, where even slight absolute deviations produce large percentage differences. Overall, while the MATLAB model offers a useful first approximation, the HYSYS simulation provides a more accurate and industry-standard estimate by incorporating non-ideal thermodynamics and rigorous property methods.

4 Economy Analysis

4.1 Capital and Operation Cost Analysis

Below is the Pie chart showing the annual operating cost distribution and the inside battery limit (ISBL) distribution:

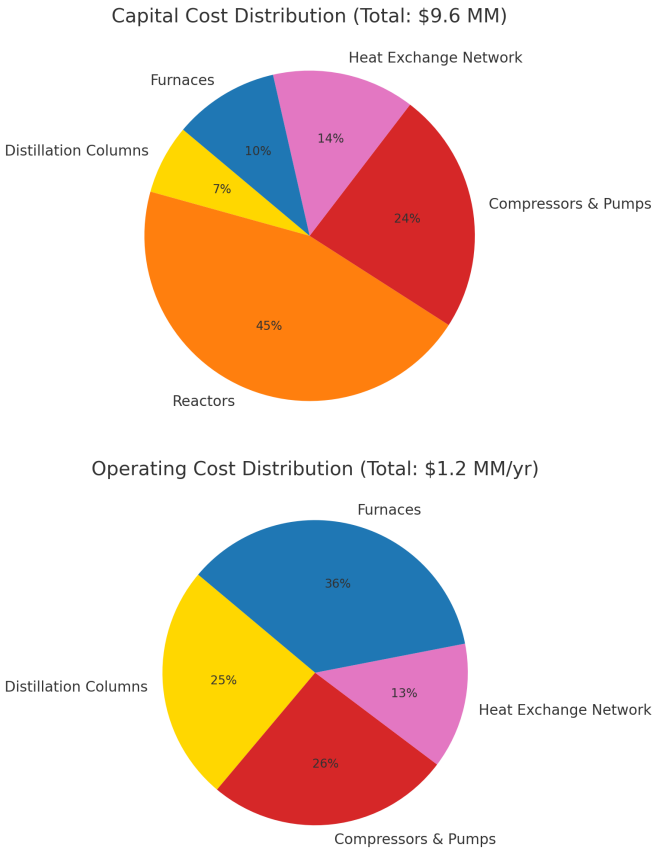


Figure 7: Top pie chart represent Capital cost distribution across major equipment groups in the HDA process. The reactor contributes the largest share (46%), followed by compressors and pumps (24%), heat exchange network (14%), furnaces (10%), and distillation columns (7%). Bottom pie char representing cost distribution excluding the reactor due to negligible operating expenses. Furnaces dominate (36%), followed by compressors and pumps (26%), distillation columns (25%), and heat exchange network (13%).

The left pie chart illustrates the **Capital Cost Distribution** for the hydrodealkylation (HDA) process, with a total installed investment of approximately $9.6 million. The reactor system accounts for the largest share of capital cost, which is justified by the requirement for a large-volume plug flow reactor (PFR) to achieve the target conversion of 99.8%. Compressors and pumps also contribute significantly due to the energy-intensive hydrogen compression and fluid transport necessary for the process. In contrast, the capital investment for distillation columns is relatively low because the reactor effluent essentially consists of benzene and biphenyl—components with starkly different volatilities that allow for efficient separation using a comparatively simple distillation unit. The heat exchange network and furnaces, while still important, represent smaller fractions of the overall capital cost.

The right pie chart presents the **Operating Cost Distribution**, totaling roughly $1.2 million per year. Furnaces incur the highest operating cost, primarily because of the continuous heat demand for maintaining the required reaction conditions, as well as the associated electricity expenses and CO_2 penalties from methane and diphenyl combustion. Compressors and pumps, critical for hydrogen compression and liquid transport, further contribute to the operational expenses. Meanwhile, the operating costs for distillation and heat exchange remain moderate, which is consistent with the straightforward nature of these units within the overall separation and energy integration strategy. Notably, the reactor is excluded from the operating cost breakdown due to its negligible running expense after installation.

4.2 Cash Flow and Internal Rate of Return Analysis

In addition to the pie chart distribution, our team has conducted a detailed cash flow analysis over the 15-year project period. Furthermore, we have calculated the Net Present Value (NPV) as a function of the project year, as illustrated in the figure below.

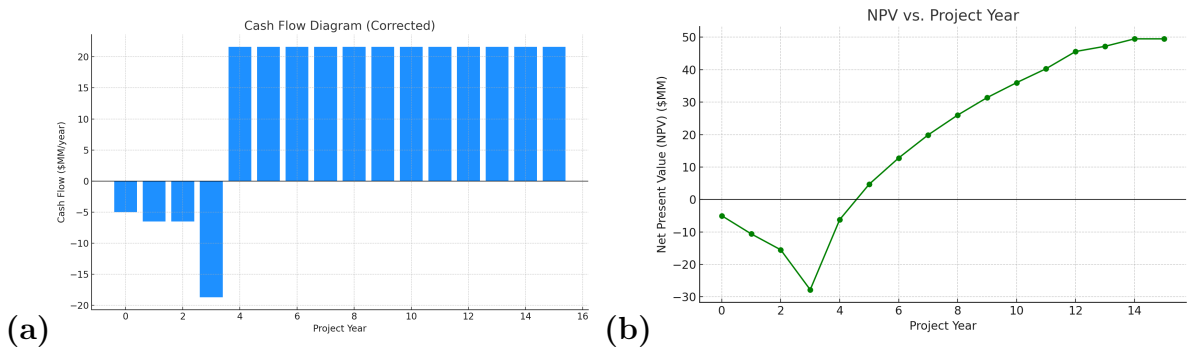


Figure 8: (a) Cash Flow Diagram illustrating the yearly cash inflows and outflows over the 15-year project duration. (b) Net Present Value (NPV) vs. Project Year, showing the cumulative value of the project over time. The NPV curve demonstrates the breakeven point and long-term profitability, with the project achieving a positive NPV after the payback period.

The cash flow diagram and net present value (NPV) analysis offer a clear assessment of the economic viability of the hydrodealkylation (HDA) process. In the initial project years (Years 0–3), the cash flow is negative due to substantial capital expenditures associated with construction, equipment installation, and start-up activities. These outflows

are expected, as the first three years are dedicated to project buildout without revenue generation. From Year 4 onward, the plant becomes operational and consistently generates positive annual cash flows of approximately $26.8 million. This sustained revenue stream reflects the reliable income from benzene production once the facility is online.

The NPV curve complements this trend, initially declining to a minimum of approximately −$27.8 million in Year 3 due to heavy upfront investment. However, it begins to rise steadily as revenue accumulates, reaching breakeven around Year 5. Beyond this point, the project yields net economic gains, ultimately achieving a final NPV of **$49.5 million** by Year 15. The internal rate of return (IRR) is **39%**, significantly exceeding typical industry hurdle rates.

These financial metrics strongly indicate that the HDA process is not only economically viable but also a highly attractive investment. Nevertheless, potential enhancements—such as improving hydrogen recovery and reducing utility costs—could further improve the project's profitability and long-term sustainability.

4.3 Sensitivity Analysis

The Design team also investigated how change in key operational factors affect the NPV, below is the tornado sensitivity analysis.

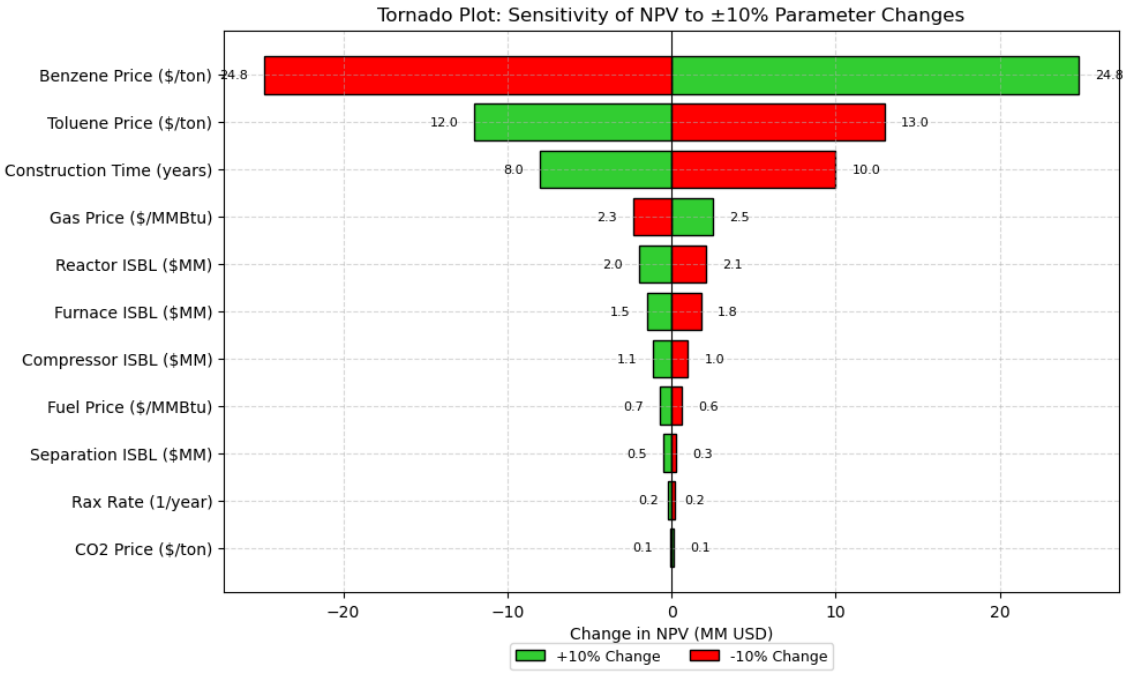


Figure 9: Tornado sensitivity analysis showing the impact of ±10% changes in key economic and operational factors on Net Present Value (NPV). Positive changes (green) and negative changes (red) highlight the sensitivity of NPV to variations in benzene price, toluene price, tax rate, operation time, and other critical parameters.

At the top of the tornado plot, the **benzene price** stands out as the most influential variable affecting the Net Present Value (NPV) of the HDA process. As the primary product, even modest fluctuations in its market price lead to substantial changes in

project profitability. A 10% increase in benzene price raises the NPV by approximately $25 million, while a 10% decrease results in a comparable reduction. This underscores the importance of price stability and highlights the value of long-term sales contracts or hedging strategies to mitigate market volatility.The **toluene price**, representing the major feedstock cost, is the second most sensitive factor. An increase in toluene cost reduces the NPV significantly, reflecting its direct impact on operating expenses and process economics. Careful procurement strategies and long-term supply agreements are therefore critical to preserving project margins.**Operation time** (meaning the years of factory running time) ranks third in sensitivity, indicating the economic importance of maximizing uptime. Longer annual operation directly improves output and revenue, while unplanned downtime or maintenance disruptions reduce profitability. Investing in plant reliability, predictive maintenance, and streamlined operations can enhance NPV by ensuring consistent process availability.

5 Process Safety and Harzards

The design and operation of the benzene production process involve various safety concerns and potential hazards that must be managed to ensure safe and efficient operation. The key risks associated with this process arise from high-temperature reactions, high-pressure conditions, and the presence of flammable and toxic chemicals. Proper hazard identification and mitigation strategies are essential to prevent accidents, equipment damage, and environmental impact.

One of the primary concerns is thermal runaway in the reactor, which can occur due to excessive feed rates or improper hydrogen-to-toluene ratio. A temperature increase beyond design limits can lead to reactor overheating, coke formation, and a risk of pressure buildup. If not controlled, this can result in catastrophic equipment failure or even explosions. To mitigate this risk, advanced temperature monitoring systems, emergency shutdown procedures, and optimized feed control loops must be implemented.

Another major hazard is hydrogen management in the gas recycle system. Hydrogen is highly flammable and can create an explosive atmosphere if leaked into the surrounding environment. Excess hydrogen purge can lead to energy inefficiencies, while insufficient hydrogen recycling can result in poor conversion and catalyst deactivation. Therefore, precise flowrate control, automated purge systems, and redundant pressure monitoring must be employed to maintain safe hydrogen handling.

The separator system poses a risk of phase separation failure, leading to product contamination and downstream equipment fouling. Poor separation efficiency can result in undesired hydrocarbons carrying over to the next stage, which affects product purity and can overload downstream distillation units. Overfilling in the separator can cause pressure build-up and possible liquid carryover into vapor systems. Regular maintenance, level alarms, and pressure control mechanisms are necessary to ensure safe operation.

Overall, implementing comprehensive hazard analysis and risk mitigation strategies is crucial to maintaining a safe and efficient process. The integration of process control measures, preventive maintenance, emergency response systems, and adherence to industry safety regulations will minimize potential hazards and enhance the reliability of the benzene production system

5.1 Chemical Hazards and Environmental Impact

The hydrodealkylation (HDA) process involves handling hazardous chemicals that pose significant safety and environmental risks. Proper containment, leak prevention, and emergency response measures are critical to ensuring process safety and minimizing environmental damage. Table 5 summarizes the primary chemical hazards, associated risks, and potential environmental impacts in the event of a leak.

Species	Flammability	Explosive Limits (V)	Toxicology	Corrosiveness
Toluene	Flammable (Flash 4°C)	UEL: 7.1%, LEL: 1.1%	CNS depression, respiratory irritation	Mild skin and eye irritant
Benzene	Highly flammable	UEL: 8.0%, LEL: 1.2%	Carcinogen, blood disorders	Toxic to aquatic life
Hydrogen	Extremely flammable	UEL: 75%, LEL: 4%	Non-toxic but asphyxiant	Non-corrosive
Methane	Flammable gas	UEL: 15%, LEL: 5%	Simple asphyxiant	Non-corrosive
Diphenyl	Combustible compound	N/A	Respiratory irritant, skin sensitizer	Bioaccumulative, low biodegradability

Table 5: Compact Chemical Hazards and Environmental Impact Analysis

Based on the information in Table 4, safety measures in the hydrodealkylation (HDA) process should prioritize containment, ventilation, and environmental protection. Hydrogen and methane pose severe flammability and explosion risks due to their high explosive limits. Proper handling procedures, leak detection systems, and inerting strategies are essential to mitigate fire and explosion hazards. Benzene and toluene are both toxic—benzene being a known carcinogen—and must be contained to prevent exposure to workers and contamination of surrounding ecosystems. Diphenyl's persistence and bioaccumulative nature indicate that spills could lead to long-term environmental harm, so secondary containment and biodegradation monitoring should be in place. Overall, this hazard analysis reinforces the need for engineering controls, PPE, and emergency response plans tailored to the properties of each chemical.

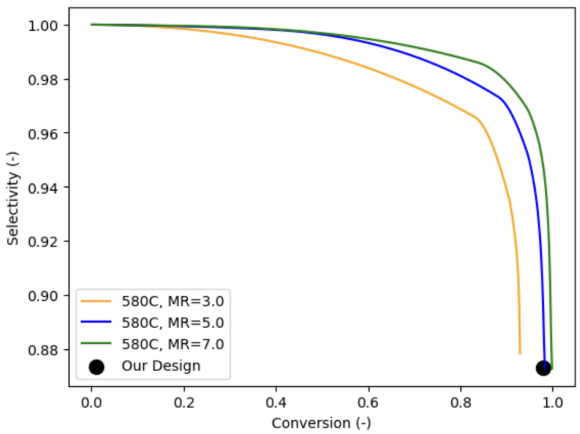
6 Process Alternatives and Next Steps

The current HDA process design includes several simplifying assumptions that warrant refinement in future development stages. For example, CO_2 emissions are estimated using a 90% efficiency factor, and all separations are assumed to be 100% efficient—both of which deviate from real-world plant behavior. Similarly, fouling and coking are not yet modeled, despite their inevitable impact on reactor performance. Future designs should incorporate realistic tray efficiencies, fouling rates, and emission factors to ensure operational accuracy and sustainability.

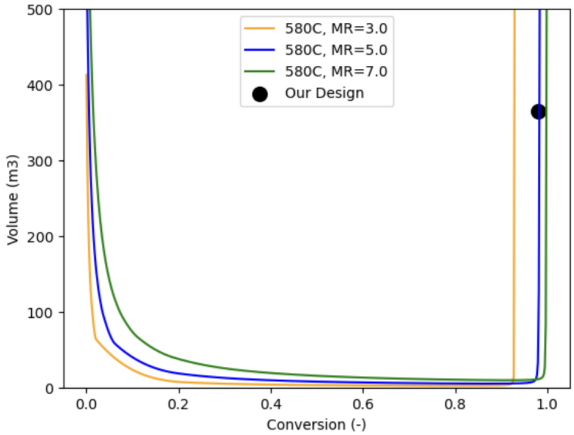
Another major limitation lies in the fixed hydrogen-to-toluene molar ratio (MR) of 5. Maintaining this ratio requires excess hydrogen, resulting in methane buildup and an

extremely high purge rate of roughly 90%. This reduces hydrogen utilization and raises feed costs. Integrating a Pressure Swing Adsorption (PSA) unit could recover hydrogen from the purge, reduce methane accumulation, and improve overall process efficiency.

In addition, MR is a critical design parameter that significantly affects both selectivity and reactor size. Figure 10 (a) illustrates that increasing MR improves selectivity but also increases hydrogen consumption. Likewise, Figure 10 (b) shows that lower MR values require significantly larger reactor volumes due to slower kinetics. Exploring MR optimization beyond the fixed value of 5 could enhance performance, economics, and sustainability.



(a) Selectivity vs. conversion at varying MR (580 °C).



(b) Reactor volume vs. conversion at varying MR (580 °C).

Figure 10: Effect of hydrogen-to-toluene molar ratio (MR) on HDA reactor performance.

7 Conclusions

The design and economic evaluation of the HDA process for benzene production demonstrate both technical feasibility and financial viability. By utilizing an adiabatic plug flow reactor, the process achieves efficient conversion of toluene into benzene while minimizing capital and operating expenses. The total capital investment is estimated at $66.5 million, with a projected net present value (NPV) of $9.1 million over a 15-year project life and an internal rate of return (IRR) of 18%. The economic analysis indicates that the process reaches its breakeven point within twelve years, confirming its long-term profitability.

Despite the favorable economic outlook, opportunities for further optimization remain. Enhancing hydrogen recovery through Pressure Swing Adsorption (PSA) could significantly reduce hydrogen losses and improve sustainability. Additionally, real-world factors such as separation inefficiencies, reactor coking, and energy losses should be incorporated into future process refinements. Implementing periodic maintenance strategies and improving separation efficiency will help maintain long-term process stability. Overall, this HDA process aligns with industry standards and provides a scalable, economically viable solution for benzene production, supporting Alkyl Products Limited's goals of cost-effective and environmentally responsible chemical manufacturing.

References

[1] ScienceDirect. Catalytic Reforming – an overview — ScienceDirect Topics. *https://www.sciencedirect.com/topics/engineering/catalytic-reforming* (accessed April 29, 2025).

[2] Grand View Research. Benzene Market Size, Share Trends Analysis Report. Available online: `https://www.grandviewresearch.com/industry-analysis/benzene-market-report` (accessed March 10, 2025).

[3] Centers for Disease Control and Prevention (CDC). Benzene: Chemical Emergency Fact Sheet. Available online: `https://www.cdc.gov/chemical-emergencies/chemical-fact-sheets/benzene.html` (accessed March 10, 2025).

[4] Doherty, M. F. Digital Design of Chemical Processes, Chapter 5: Reactor and Recycle System; Draft Edition, 2024.

[5] Chada, J.; Doherty, M. F. FEL-1 Techno-Economic Assessment of a Process for Benzene Production. *Design Project: Benzene Production by Hydrodealkylation (HDA)*, Department of Chemical Engineering, UC Santa Barbara, Winter 2025.

[6] Jiang, B.; Xu, Y.; Zhang, Y.; Yu, H.; Wang, L.; Chen, X. Separation Performance of Pressure Swing Adsorption for Hydrogen Purification. *Energy Procedia*, 2017, **142**, 2487–2493. doi:10.1016/j.egypro.2017.12.086.

[7] Taiyo Gas. Sources, Effects, and Uses of Methane Gas. Available online: *https://www.taiyugas.com/news/sources-effects-and-uses-of-methane-gas.html* (accessed March 10, 2025).

[8] National Institute of Standards and Technology (NIST). Toluene, NIST Chemistry WebBook. Available online: `https://webbook.nist.gov/cgi/cbook.cgi?ID=108-88-3` (accessed March 10, 2025).

[9] National Institute of Standards and Technology (NIST). Benzene, NIST Chemistry WebBook. Available online: `https://webbook.nist.gov/cgi/inchi?ID=C71432&Mask=1` (accessed March 10, 2025).

[10] National Institute of Standards and Technology (NIST). Hydrogen, NIST Chemistry WebBook. Available online: `https://webbook.nist.gov/cgi/cbook.cgi?ID=C1333740` (accessed March 10, 2025).

[11] National Institute of Standards and Technology (NIST). Diphenyl, NIST Chemistry WebBook. Available online: `https://webbook.nist.gov/cgi/cbook.cgi?ID=92-52-4` (accessed March 10, 2025).

[12] National Institute of Standards and Technology (NIST). Methane, NIST Chemistry WebBook. Available online: `https://webbook.nist.gov/cgi/cbook.cgi?ID=74-82-8` (accessed March 10, 2025).

[13] Doherty, M. F. Equipment Cost Correlations & Related Data. *Digital Design of Chemical Processes*, 2025 Edition.

[14] Doherty, M. F.; Malone, M. F. *Conceptual Design of Distillation Systems*; McGraw-Hill: New York, 2001.

8 Appendix

8.1 Appendix A: Detailed Mole Balance for HDA Process

8.1.1 Reactor Model Setup

The plug flow reactor (PFR) is modeled by a system of ordinary differential equations (ODEs) that describe the molar flow rate of each species F_i (mol/s) as a function of reactor volume V (L):

$$\frac{dF_T}{dV} = -r_1, \tag{3}$$

$$\frac{dF_H}{dV} = -r_1 + r_2, \tag{4}$$

$$\frac{dF_B}{dV} = +r_1 - 2r_2, \tag{5}$$

$$\frac{dF_M}{dV} = +r_1, \tag{6}$$

$$\frac{dF_D}{dV} = +r_2, \tag{7}$$

where r_1 and r_2 are the rates of the primary and side reactions, respectively, expressed in mol/L/s.

The reactions are:

$$\text{Primary:} \quad \mathrm{T} + \mathrm{H_2} \longrightarrow \mathrm{B} + \mathrm{M}, \quad \text{Side:} \quad 2\,\mathrm{B} \rightleftharpoons \mathrm{D} + \mathrm{H_2}.$$

8.1.2 Basis Simulation (1 mol/s Toluene)

We set a basis of 1 mol/s toluene at the reactor inlet:

$$F_{T,0} = 1.0 \text{ mol/s}, \tag{8}$$

$$F_{H,0} = \mathrm{MR} \times F_{T,0} \quad (\mathrm{MR} = 5.0), \tag{9}$$

$$F_{M,0} = \frac{7}{95} \times F_{H,0}, \tag{10}$$

$$F_{B,0} = 0, \quad F_{D,0} = 0. \tag{11}$$

The reactor is simulated under specified temperature and pressure conditions, and the ODEs are integrated over a large reactor volume to ensure the system approaches equilibrium.

8.1.3 Selectivity and Conversion

At each volume point, the following quantities are calculated:

$$S = \frac{F_B}{F_{T,0} - F_T}, \quad \text{(benzene selectivity)}, \tag{12}$$

$$x = \frac{F_{T,0} - F_T}{F_{T,0}}, \quad \text{(toluene conversion)}. \tag{13}$$

where F_B and F_T are the benzene and toluene molar flow rates at the reactor outlet.

8.1.4 Scaling to Desired Production Rate

Given a target benzene production rate P_B (mol/s), the required true reactor inlet flows are scaled as:

$$F_T^{\text{true}} = \frac{P_B}{S \times x}, \tag{14}$$

$$F_H^{\text{true}} = \text{MR} \times F_T^{\text{true}}, \tag{15}$$

$$F_M^{\text{true}} = \frac{7}{95} \times F_H^{\text{true}}, \tag{16}$$

$$F_B^{\text{true}} = 0, \quad F_D^{\text{true}} = 0. \tag{17}$$

The total inlet molar flow rate is:

$$F_{\text{tot}}^{\text{true}} = F_T^{\text{true}} + F_H^{\text{true}} + F_M^{\text{true}}.$$

8.1.5 True Reactor Outlet Profiles

The true scaled inlet stream ($F_T^{\text{true}}, F_H^{\text{true}}, F_M^{\text{true}}$) is then integrated again through the PFR model under the same temperature and pressure to obtain the true reactor outlet molar flow rates $F_T^{\text{out}}, F_H^{\text{out}}, F_M^{\text{out}}, F_B^{\text{out}}, F_D^{\text{out}}$ and outlet temperature.

8.1.6 Plant-Level Mass Balances: Recycle and Purge

After obtaining the true reactor outlet profiles, the recycle and purge system is defined as follows.

The moles of methane generated inside the reactor are:

$$F_M^{\text{gen}} = F_M^{\text{out}} - \left(F_T^{\text{true}} \times \frac{5 \times 7}{95} \right). \tag{18}$$

*A.6.2 Purge Fraction The gas-phase purge fraction is calculated based on methane:

$$\text{Purge fraction} = \frac{F_M^{\text{fresh}} + F_M^{\text{gen}}}{F_M^{\text{out}}}, \tag{19}$$

where F_M^{fresh} is the fresh methane feed rate.

*A.6.3 Fresh Hydrogen Feed The fresh hydrogen feed rate is solved by:

$$F_H^{\text{fresh}} = \frac{5F_T^{\text{true}} - F_H^{\text{out}} \left(1 - \frac{F_M^{\text{gen}}}{F_M^{\text{out}}}\right)}{1 - \frac{5}{95} \left(\frac{F_H^{\text{out}}}{F_M^{\text{out}}}\right)}. \tag{20}$$

*A.6.4 Fresh Methane Feed The fresh methane feed rate is linked to fresh hydrogen feed:

$$F_M^{\text{fresh}} = \frac{0.05}{0.95} \times F_H^{\text{fresh}}. \tag{21}$$

*A.6.5 Fresh Toluene Feed The fresh toluene feed is:

$$F_T^{\text{fresh}} = F_T^{\text{true}} - F_T^{\text{out}}. \tag{22}$$

*A.6.6 Recycle and Purge Flows The recycle and purge molar flow rates for each component are:

$$F_i^{\text{rec}} = (1 - \text{purge fraction}) \times F_i^{\text{out}}, \tag{23}$$
$$F_i^{\text{purge}} = \text{purge fraction} \times F_i^{\text{out}}, \tag{24}$$

where i represents toluene (T), hydrogen (H_2), or methane (CH_4).

8.1.7 Outlet Stream Destination

Below is a table outlining the decisions made regarding the destination of each component in the reactor outlet stream. These decisions were based on optimizing process efficiency, minimizing waste, and maximizing economic and environmental benefits.

Component	Destination	Justification
Benzene (C_6H_6)	Product Stream	Primary desired product, sent to separation and purification for sale.
Methane (CH_4)	Purge Stream	Inert byproduct with no further use in the process, needs removal to prevent buildup. CH4 can be collect to be burned as fuels
Toluene (C_7H_8)	Recycle Stream	Unreacted toluene is separated and recycled to improve efficiency and reduce raw material costs.
Diphenyl ($C_{12}H_{10}$)	Fuel Stream	Used as a fuel to recover energy and improve process sustainability.
Hydrogen (H_2)	Recycle/Purge Stream	Recycled for process efficiency; excess is purged to maintain optimal reactor operation.

Table 6: Destination of Each Component in the Reactor Outlet Stream

8.2 Appendix B: Reaction Kinetics and Thermodynamics

8.2.1 Reaction Kinetics[2]

Diphenyl is to be considered as a placeholder for a collection of by-products formed by reversible chemical reactions. Therefore, there is no market for this "substance," and its only value is to burn it as fuel in the process. The reactions take place at high temperature and high pressure without a catalyst (homogeneous gas phase reaction).

The following concentration-based rate equation was used to model the kinetics of the primary reaction for benzene production. The rate of the primary reaction, ξ_1, is in units gmol/(L h), with the rate expression:

$$\xi_1 = k_{1,J} C_T C_H^{0.5} \tag{25}$$

where $k_{1,J}$ is the forward rate constant; C_T and C_H are the molar concentrations of toluene and hydrogen, respectively, in gmol/L. The forward rate constant is modeled with the Arrhenius equation and given by:

$$k_{1,J} = k_{1,0} \exp\left(\frac{-E_{a1}}{RT}\right) \tag{26}$$

where the prefactor $k_{1,0} = 1.188 \times 10^{14}$ with units $\left(\sqrt{\frac{L}{gmol}}\frac{1}{h}\right)$, activation energy $E_{a1} = 52.0$ kcal/gmol, and T is the temperature in Kelvin.

A partial pressure-based rate equation is used to model the kinetics of the side-reaction that produces diphenyl. We have written this model in terms of the total pressure, P, and mole fractions y. The rate of the side-reaction, ξ_2, is in units gmol/(L h), with the rate expression:

$$\xi_2 = k_{2,J} P^2 \left(y_B^2 - \frac{y_D y_H}{K_{eq}}\right) \tag{27}$$

where $k_{2,J}$ is the forward rate constant of the side reaction, P is the reactor pressure in Pascal, K_{eq} is the dimensionless reaction equilibrium rate constant; y_B, y_D, y_H are the gas phase mole fractions of benzene, diphenyl, and hydrogen, respectively. The forward reaction rate constant $k_{2,J}$ of the side-reaction is modeled as follows:

$$k_{2,J} = k_{2,0} \exp\left(\frac{-E_{a2}}{RT}\right) \tag{28}$$

where $k_{2,0} = 1.1647 \times 10^{-5}$ gmol/(L Pa^2 h) and $E_{a2} = 30.19$ kcal/gmol, and T is the temperature in Kelvin. The dimensionless equilibrium constant for the reaction is given by:

$$\log_{10} K_{eq}(T) = \frac{734.92}{T} - 8.1046 + 3.1294 \log_{10} T - 7.0804 \times 10^{-4} T + 8.523 \times 10^{-8} T^2 \tag{29}$$

The reaction equilibrium equation for this reaction is given by:

$$K_{eq} = \frac{y_D y_H}{y_B^2} \tag{30}$$

The reactor temperature must be maintained within a specific range to prevent operational inefficiencies and equipment damage. The upper temperature limit is 650°C as exceeding this value can cause coke formation, leading to carbon deposits on reactor walls and equipment. This constraint applies whether the reactor is operated adiabatically. Conversely, a lower temperature limit of 550°C is necessary to sustain a sufficiently fast reaction rate; below this threshold, conversion rates are too slow. The

reactor operates within a pressure range of 30–34 bar. The rate expressions provided are valid for these temperature and pressure conditions. Experimental studies in laboratory reactors have established an optimal hydrogen-to-toluene molar ratio of 5:1 at the reactor inlet. This ratio is essential for developing accurate rate expressions. While the kinetic model may remain valid for slightly different molar ratios, any significant deviation requires validation through additional R&D studies to justify an economic incentive for adjustments.

8.2.2 Reaction Thermodynamics

The hydrodealkylation (HDA) process involves multiple components, each with unique thermophysical properties affecting reaction kinetics and energy balance. Table 7 summarizes the molecular weights and standard heats of formation for key species.

Component	**Molecular Weight (g/mol)**	**Heat of Formation (kJ/mol)**
Toluene (T)[**5**]	92.14	12.0
Hydrogen (H_2)[**7**]	2.02	0.0
Benzene (B)[**6**]	78.11	49.0
Methane (M)[**8**]	16.04	-74.8
Diphenyl (D)[**9**]	154.21	174.3

Table 7: Thermodynamic properties of species in the HDA process

The enthalpy changes of the two main reactions in the HDA process are determined using the standard heats of formation:

- **Primary Reaction:** Toluene + Hydrogen → Benzene + Methane

$$\Delta H_1 = (H_f^B + H_f^M) - (H_f^T + H_f^H) \tag{31}$$

$$\Delta H_1 = (49.0 + (-74.8)) - (12.0 + 0.0) = -37.8 \text{ kJ/mol} \tag{32}$$

- **Side Reaction:** 2 Benzene → Diphenyl + Hydrogen

$$\Delta H_2 = (H_f^H + H_f^D) - 2H_f^B \tag{33}$$

$$\Delta H_2 = (0.0 + 174.3) - 2(49.0) = 76.3 \text{ kJ/mol} \tag{34}$$

The temperature-dependent molar heat capacity $C_{p,i}(T)$ of species i is calculated using the DIPPR 101 form, which differs from the polynomial approximation previously reported. The correct expression is:

$$C_{p,i}(T) = C_1 + C_2 \left(\frac{C_3/T}{\sinh(C_3/T)} \right)^2 + C_4 \left(\frac{C_5/T}{\cosh(C_5/T)} \right)^2 \tag{35}$$

where C_1, C_2, C_3, C_4, C_5 are DIPPR-specific heat capacity coefficients for each species. The result is in units of J/(kmol·K), and is divided by 1000 to obtain J/(mol·K).

Table 8: DIPPR 101 Heat Capacity Coefficients for HDA Species

Component	C_1	C_2	C_3	C_4	C_5
Hydrogen (H_2)	27,617	9,560	2.466	3,760	567.6
Methane (CH_4)	33,298	79,933	2.0869	41,602	991.96
Toluene (C_7H_8)	58,140	286,300	1.4406	189,800	650.43
Benzene (C_6H_6)	44,767	230,850	1.4792	168,360	677.66
Diphenyl ($C_{12}H_{10}$)	107,590	421,050	1.9041	417,850	828.81

To compute the temperature-dependent molar enthalpy $H_i(T)$, the heat capacity function is integrated from a reference temperature $T_{\text{ref}} = 298.15\,\text{K}$ to the target temperature T:

$$H_i(T) = \Delta H_{f,i} + \int_{T_{\text{ref}}}^{T} C_{p,i}(T')\,dT' \tag{36}$$

where $\Delta H_{f,i}$ is the standard formation enthalpy of species i in J/mol.

The reaction enthalpies for the two main reactions are computed as follows:

$$\Delta H_1(T) = H_B(T) + H_M(T) - [H_T(T) + H_H(T)] \tag{37}$$
$$\Delta H_2(T) = H_D(T) + H_H(T) - 2H_B(T) \tag{38}$$

This corrected algorithm captures the nonlinear behavior of C_p with respect to temperature and results in a more accurate enthalpy change calculation compared to the previous polynomial approximation.

8.3 Appendix C: Estimation of the Underwood binary distillation[14]

8.3.1 Calculation of Relative Volatility

We set the system pressure to 1 bar (equivalent to 760 mmHg), and specify the operating temperature T (°C).

The saturation pressures of components A and B are calculated using the Antoine equation:

$$\log_{10}\left(P_A^{\text{sat}}\right) = A_A - \frac{B_A}{T + C_A} \tag{39}$$

$$\log_{10}\left(P_B^{\text{sat}}\right) = A_B - \frac{B_B}{T + C_B} \tag{40}$$

where A_i, B_i, C_i are Antoine coefficients for component i.

The vapor-liquid equilibrium constants (K-values) for each component are then determined by:

$$K_A = \frac{P_A^{\text{sat}}}{P_{\text{total}}} \tag{41}$$

$$K_B = \frac{P_B^{\text{sat}}}{P_{\text{total}}} \tag{42}$$

After calculating the K-values, the component with the lower volatility (smaller K) is selected as the reference component.

Finally, the relative volatility α of the more volatile component relative to the reference is given by:

$$\alpha = \frac{K_{\text{more volatile}}}{K_{\text{reference}}} \tag{43}$$

8.3.2 Reflux Ratio and Boil-Up Ratio Calculation

In this simulation, we apply the Underwood method assuming **direct distillation**, meaning the light key component is recovered at the top of the column (in the distillate).

First, we solve for the Underwood root θ by satisfying the following relation using feed composition:

$$\sum_i \frac{\alpha_i z_i}{\alpha_i - \theta} = 1 \tag{44}$$

where:

- α_i is the relative volatility of component i,
- z_i is the mole fraction of component i in the feed.

Once θ is determined, the **minimum reflux ratio** for a **direct split** is calculated as:

$$r_{\min} + 1 = \frac{\alpha_A x_{A,D}}{\alpha_A - \theta} + \frac{\alpha_B x_{B,D}}{\alpha_B - \theta} \tag{45}$$

For a **indirect split** (where the heavy key is the target at the bottom), the **minimum boil-up ratio** is calculated using:

$$s_{\min} = \frac{\alpha_A x_{A,B}}{\theta - \alpha_A} + \frac{\alpha_B x_{B,B}}{\theta - \alpha_B} \tag{46}$$

In this simulation, because a **direct split** is assumed, we only use the reflux ratio equation.

The actual operating reflux ratio is selected by:

$$r = 2 \times r_{\min} \tag{47}$$

The operational boil-up ratio is then determined based on the material balance:

$$s = \frac{D}{B}(r + q) - (1 - q) \tag{48}$$

where:

- D and B are the distillate and bottoms flow rates,
- q is the thermal condition of the feed. In this case, $q = 1$ (saturated liquid feed).

To fully define the rectifying and stripping sections, we must solve for Underwood roots ϕ separately for each section:

- **Rectifying section roots** are obtained by solving:

$$\sum_i \frac{\alpha_i x_{D,i}}{\alpha_i - \phi} = r + 1 \tag{49}$$

- **Stripping section roots** are obtained by solving:

$$\sum_i \frac{\alpha_i x_{B,i}}{\alpha_i - \phi} = s \tag{50}$$

These nonlinear equations are solved numerically (e.g., using the bisection method) to find two valid roots for each section, corresponding to the two key components.

The roots obtained are then used to calculate the number of theoretical stages in the rectifying and stripping sections separately.

8.3.3 Calculation of Rectifying and Stripping Stages

After determining the operational reflux and boil-up ratios, we solve for new Underwood roots based on the distillate and bottoms compositions.

The roots for the rectifying and stripping sections are determined by solving the following equations:

- **Rectifying section roots** (using distillate composition $x_{D,i}$):

$$\sum_i \frac{\alpha_i x_{D,i}}{\alpha_i - \phi} = r + 1 \tag{51}$$

- **Stripping section roots** (using bottoms composition $x_{B,i}$):

$$\sum_i \frac{\alpha_i x_{B,i}}{\alpha_i - \phi} = s \tag{52}$$

The above non-linear equations are numerically solved to find two valid roots for each section: ϕ_1 and ϕ_2.

Since the feed condition is saturated liquid ($q = 1$), the feed stage composition $x_{i,f}$ equals the feed mole fraction z_i:

$$x_{i,f} = z_i \tag{53}$$

This is because there is no enthalpy change between the feed and the liquid operating line under saturated liquid conditions.

The number of theoretical stages in the rectifying section N_T is calculated by:

$$N_T = \frac{\ln\left(\frac{\sum_i \frac{\alpha_i x_{f,i}}{\alpha_i - \phi_2}}{\sum_i \frac{\alpha_i x_{f,i}}{\alpha_i - \phi_1}}\right)}{\ln\left(\frac{\phi_1}{\phi_2}\right)} \tag{54}$$

Similarly, the number of theoretical stages in the stripping section N_B is calculated by:

$$N_B = \frac{\ln\left(\frac{\sum_i \frac{\alpha_i x_{f,i}}{\alpha_i - \phi_1}}{\sum_i \frac{\alpha_i x_{f,i}}{\alpha_i - \phi_2}}\right)}{\ln\left(\frac{\phi_1}{\phi_2}\right)} \tag{55}$$

where:

- ϕ_1 and ϕ_2 are the two roots obtained for each section,
- $x_{f,i}$ is the feed stage mole fraction.

The equivalent total number of theoretical stages N_{eq} is the sum:

$$N_{\text{eq}} = N_T + N_B + 1 \tag{56}$$

Finally, the real number of stages N_{real} is calculated by considering a stage efficiency E_0:

$$N_{\text{real}} = \left\lceil \frac{N_{\text{eq}}}{E_0} \right\rceil \tag{57}$$

8.3.4 Vapor Flowrates and Energy Balances

The vapor flowrates at the top and bottom of the column are calculated based on the material balance:

$$V_T = (r+1)D \tag{58}$$

$$V_B = sB \tag{59}$$

where:

- V_T is the vapor flowrate in the rectifying section,
- V_B is the vapor flowrate in the stripping section,
- r is the operational reflux ratio,
- s is the operational boil-up ratio,
- D and B are the distillate and bottoms flowrates, respectively.

Because the feed is assumed to be a **saturated liquid** ($q = 1$), there is no enthalpy imbalance at the feed stage. Thus, we assume:

$$V_T \approx V_B \tag{60}$$

in the column.

The heat duties for the condenser and reboiler are calculated by assuming:

- Constant heat capacities (C_p) for vapor and liquid streams,
- The distillate and bottoms are pure components, meaning each product is considered composed of a single major species.

The condenser duty Q_C is given by:

$$Q_C = \lambda_D \times V_T \tag{61}$$

The reboiler duty Q_R is given by:

$$Q_R = \lambda_B \times V_B \tag{62}$$

where:

- λ_D is the latent heat of vaporization of the distillate component (e.g., Toluene or Benzene),
- λ_B is the latent heat of vaporization of the bottoms component (e.g., Diphenyl or Toluene),
- Heat duties Q_C and Q_R are in units of kJ/s.
- The feed is a saturated liquid ($q = 1$),
- Constant latent heats λ_D and λ_B are assumed,
- The distillate and bottoms streams are considered pure components.

8.3.5 Cross-Sectional Area Calculation

The cross-sectional area A of the column is determined based on the vapor flowrates in the rectifying and stripping sections.

The area required for vapor handling is calculated by:

$$A = \frac{M_v}{\sqrt{\rho_l \, \rho_v} \, \phi_{\text{flood}} \, c_0} \times \frac{V}{A_n/A} \tag{63}$$

where:

- M_v is the molecular weight of the vapor phase (g/mol),
- ρ_l is the liquid density (g/m^3),
- ρ_v is the vapor density (g/m^3),
- ϕ_{flood} is the design fraction of flooding velocity (typically 0.6),
- c_0 is the vapor capacity constant (depends on tray spacing, from Fair's correlation),
- A_n/A is the active tray area fraction,
- V is the vapor flowrate (mol/s).

We assume a **tray spacing of 24 inches** (0.61 m). According to Fair's correlation, the constants used are:

Tray Spacing	12 in	18 in	24 in
c_0 (m/h)	252	329	439
c_0 (ft/h)	828	1080	1440
c_1	2.0	2.3	2.5
c_2	1.0	1.1	1.2

Table 9: Constants for Fair's Correlation

Thus, for 24-inch tray spacing, we use:

$$c_0 = 439 \, \text{m/h}$$

Thus, the areas are calculated separately:

- **Rectifying Section Area:**

$$A_{\text{rect}} = \frac{M_{v,\text{rect}}}{\sqrt{\rho_{l,\text{rect}} \rho_{v,\text{rect}}} \, \phi_{\text{flood}} \, c_0} \times \frac{V_T}{A_n/A}$$

  where properties correspond to the distillate component.

- **Stripping Section Area:**

$$A_{\text{strip}} = \frac{M_{v,\text{strip}}}{\sqrt{\rho_{l,\text{strip}} \rho_{v,\text{strip}}} \, \phi_{\text{flood}} \, c_0} \times \frac{V_B}{A_n/A}$$

where properties correspond to the bottoms component.

8.3.6 Column Diameter and Height Estimation

The column diameter for each section is computed from the cross-sectional area using the following geometric relation:

$$D = \sqrt{\frac{4A}{\pi}} \tag{64}$$

where:

- A is the cross-sectional area in m^2,
- D is the internal column diameter in meters.

This is applied separately for the rectifying and stripping sections using their respective area values.

We assume that the column is designed with a **tray spacing of 24 inches**, which is equivalent to:

$$H_t = 24\,\mathrm{in} = 0.6096\,\mathrm{m} \tag{65}$$

To ensure proper liquid-vapor disengagement, the column must also include an additional allowance for headroom above the top tray and below the bottom tray. We assume a minimum clearance equal to 3 tray spacings:

$$H_{\min} = 3H_t \tag{66}$$

The total height of the column is calculated by combining the number of real stages N_{real} and the assumed tray spacing:

$$H = H_{\min} + N_{\mathrm{real}} \times H_t \tag{67}$$

The overall column height is then divided between the rectifying and stripping sections based on their respective theoretical stage counts:

$$H_{\mathrm{rect}} = \frac{N_T}{N_{\mathrm{eq}}} \times H \qquad \text{and} \qquad H_{\mathrm{strip}} = \frac{N_B}{N_{\mathrm{eq}}} \times H \tag{68}$$

where:

- N_T and N_B are the number of theoretical stages in the rectifying and stripping sections,
- $N_{\mathrm{eq}} = N_T + N_B$ is the total theoretical stage count,
- H is the total column height,
- H_{rect}, H_{strip} are the heights allocated to each section.

8.3.7 Calculation of the Heating Exchange Area for Condenser and Reboiler

The required heat exchanger area for both the condenser and reboiler is calculated using the fundamental heat transfer equation:

$$Q = U A_h \Delta T_{\text{avg}} \tag{69}$$

where:

- Q is the heat duty in watts (W),
- U is the overall heat transfer coefficient in W/m^{2}K,
- A_h is the heat transfer area in m^2,
- ΔT_{avg} is the average temperature difference across the exchanger in $^\circ C$.
- The average temperature difference is assumed to be within the typical range:

$$\Delta T_{\text{avg}} = 50 - 60^\circ C$$

- The heat transfer coefficient is chosen from Table 9:

Table 10: Typical heat transfer coefficients, U, for shell and tube exchangers in W/m^{2}K.

	Hot Side Fluid	Cold Side Fluid						
		1 Cooling Water	**2** Low Visc. Org. Liq.	**3** High Visc. Org. Liq.	**4** Boiling Water	**5** Boiling Org.	**6** Low P Gas	**7** High P Gas
1	Low Viscosity Organic Liquid	800	550	150	700	550	100	375
2	High Viscosity Organic Liquid	140	130	80	140	130	65	120
3	Condensing Steam (no dissolved air)	1600	820	170	1430	820	110	530
4	Condensing Steam (with dissolved air)	1440	775	167	1300	775	100	475
5	Cond. Hydrocarbon (no inert gas)	770	530	160	720	530	100	390
6	Cond. Hydrocarbon (10% inert gas)	350	280	220	330	280	85	240
7	Low Pressure Gas (∼1 bar)	105	100	70	105	100	55	95
8	High Pressure Gas (∼20 bar)	485	375	140	470	375	95	300

The required areas for the condenser and reboiler are then calculated as:

$$A_C = \frac{Q_C}{U\,\Delta T_{\text{avg}}} \qquad \text{and} \qquad A_R = \frac{Q_R}{U\,\Delta T_{\text{avg}}} \tag{70}$$

These areas are computed over a range of ΔT_{avg} values (typically from $50^\circ C$ to $60^\circ C$) to analyze sensitivity and select conservative exchanger sizes.

8.4 Appendix D: Capital Cost of Process Equipment[12]

8.4.1 Process Furnaces

Mid–1968 cost, box or A–frame construction with multiple tube banks, field erection.

$$\text{Purchased Cost} = \left(\frac{\text{CEPCI}}{113.7}\right) \times (5.52 \times 10^3) Q^{0.85} F_c \tag{71}$$

where Q = absorbed duty (in 10^6 Btu/hr), with the range: $20 < Q < 300$, and:

$$F_c = F_d + F_m + F_p \tag{72}$$

Table 11: Correction Factors F_c for Process Furnace

Design Type Design Pressure (psi)	F_d F_p	Radiant Tube Material	F_m
Process Heater	1.00	Carbon Steel	0.0
Up to 500	0.00		
Pyrolysis	1.10	Chrome/Moly	0.35
1,000	0.10		
Reformer (no catalyst)	1.35	Stainless	0.75
1,500	0.15		
2,000	0.25		
2,500	0.40		
3,000	0.60		

The installed cost (ISBL) is given by:

$$\text{Installed Cost} = \left(\frac{\text{CEPCI}}{113.7}\right) \times (5.52 \times 10^3) Q^{0.85} (1.27 + F_c) \tag{73}$$

8.4.2 Direct-Fired Heaters

Mid–1968 cost, cylindrical construction, field erection.

$$\text{Purchased Cost} = \left(\frac{\text{CEPCI}}{113.7}\right) \times (5.07 \times 10^3) Q^{0.85} F_c \tag{74}$$

where Q = absorbed duty (in 10^6 Btu/hr), with the range: $2 < Q < 30$.

Table 12: Correction Factors F_c for Direct-Fired Heaters

Design Type Design Pressure (psi)	F_d F_p	Radiant Tube Material	F_m
Cylindrical	1.00	Carbon Steel	0.0
Up to 500	0.00		
Dowtherm	1.33	Chrome/Moly	0.45
1,000	0.15		
		Stainless	0.50
1,500	0.20		

The installed cost (ISBL) is given by:

$$\text{Installed Cost} = \left(\frac{\text{CEPCI}}{113.7}\right) \times (5.07 \times 10^3) Q^{0.85} (1.23 + F_c) \tag{75}$$

8.4.3 Heat Exchangers

Mid–1968 cost, shell and tube, complete fabrication.

$$\text{Purchased Cost} = \left(\frac{\text{CEPCI}}{113.7}\right) \times (101.3 A^{0.65} F_c) \tag{76}$$

where A = area (ft^2), with the range: $200 < A < 25,000$.

$$F_c = (F_d + F_p) F_m \tag{77}$$

Table 13: Correction Factors for Heat Exchangers

Design Type	F_d	Design Pressure (psi) F_p
Kettle Reboiler	1.35	Up to 150 0.0
Floating Head	1.00	300 0.10
U-Tube	0.85	400 0.25
Fixed-Tube Sheet	0.80	800 0.52
		1,000 0.55

The installed cost (ISBL) is given by:

$$\text{Installed Cost} = \left(\frac{\text{CEPCI}}{113.7}\right) \times (101.3 A^{0.65})(2.29 + F_c) \tag{78}$$

8.4.4 Gas Compressors

Mid–1968 cost, centrifugal machine, motor drive, base plate, and coupling.

$$\text{Purchased Cost} = \left(\frac{\text{CEPCI}}{113.7}\right) \times (517.5)(bhp)^{0.82} F_c \tag{79}$$

where bhp = brake horsepower; range: $30 < bhp < 10,000$.

$$F_c = F_d \tag{80}$$

The installed cost (ISBL) is given by:

$$\text{Installed Cost} = \left(\frac{\text{CEPCI}}{113.7}\right) \times (517.5)(bhp)^{0.82}(2.11 + F_c) \tag{81}$$

8.4.5 Pressure Vessels, Columns, and Reactors

$$\text{Purchased Cost} = \left(\frac{\text{CEPCI}}{113.7}\right) \times (101.9 D^{1.066} H^{0.82} F_c) \tag{82}$$

where D = diameter (ft), H = height (ft).

8.4.6 Separation Cost

- $W_{\min}$ – Minimum energy requirement for separation
- λ – Dimensionless scale factor (range: 20–50) used to determine actual energy
- F_1, F_2 – Streams in the separation system
- $OPEX$ – Annual separation cost
- $C_{\$/W}$ – Cost per watt
- W_{real} – Actual energy consumption in watts
- CAPEX (ISBL) – Inside Battery Limits capital expenditure
- R – Universal gas constant
- T – Absolute temperature

$$W_{\min} = F_1 RT \left\{1 \times \ln\left(\frac{1}{0.6}\right) + 0\right\} + F_2 RT \left\{0 + 1 \times \ln\left(\frac{1}{0.4}\right)\right\} \tag{83}$$

$$\text{OPEX} = \varepsilon \lambda W_{\min} \tag{84}$$

$$\text{CAPEX (ISBL)} = C_{\$/W} \times W_{\text{real}} \tag{85}$$

8.4.7 Distillation Column Trays and Tower Internals

Installed Cost (ISBL), \$:

$$\text{ISBL} = \left(\frac{\text{CEPCI}}{113.7}\right) \cdot 4.7 D^{1.55} H F_c \tag{D.11}$$

Where D = diameter, ft, H = tray stack height, ft (24-in. spacing)

$$F_c = F_s + F_t + F_m$$

Table 14: Correction factors for column trays.

Tray Spacing, inches	24	18	12	**Tray Type**	F_t	**Tray Material**
F_s	1.0	1.4	2.2	Grid (no downcomer)	0.0	CS ($F_m = 0.0$)
				Plate / Sieve	0.0	SS ($F_m = 1.7$)
				Trough or Valve	0.4	Monel ($F_m = 8.9$)
				Bubble Cap	1.8	
				Koch Kascade	3.9	

8.5 Appendix E: Economic Assumption, Formular and Spreadsheet

8.5.1 Economic Set UP

The economic feasibility of the HDA process is assessed using key financial metrics, including Total Capital Investment (TCI), Net Present Value (NPV), and Internal Rate of Return (IRR). Table 15 provides the financial parameters necessary for the assessment of profitability.

Profitability Metric	**Value**
Enterprise Rate	15% per year
Total tax rate (combined federal, state, and local taxes)	27% per year
Admin. & General Services (AGS)	5% of annual revenues
Construction period	3 years
Period of Plant Operation	12 years
Project life (Construction + Operation)	15 years
Depreciation schedule	10-year linear
CEPCI for the year 2025	800
Salvage value	5% of fixed capital investment

Table 15: Profitability Metrics for the Project[**2**]

The following assumptions are made in this analysis:

- **Enterprise Rate:** The expected return on investment is set at **15% per year**.
- **Tax Rate:** A total tax rate of **27% per year** includes federal, state, and local taxes.
- **Project Lifespan:** The project has a **3-year construction period** followed by **12 years of operation**, leading to a total of **15 years**.
- **Depreciation:** A **10-year linear depreciation** method is used for capital recovery.
- **CEPCI Index:** The **Chemical Engineering Plant Cost Index (CEPCI) for 2025** is assumed to be **800**.
- **Salvage Value:** At the end of the project life, the salvage value is assumed to be **5% of the fixed capital investment**.

The total capital investment (TCI) consists of fixed capital investment (FCI) and working capital (WC):

$$TCI = FCI + WC \tag{86}$$

where:

- **FCI:** Represents the cost of plant construction and equipment.

- **WC:** Accounts for startup and operational costs before revenue generation.

Net present value (NPV) is a key indicator of project profitability and is calculated as:

$$NPV = \sum_{t=0}^{n} \frac{C_t}{(1+r)^t} - C_0 \tag{87}$$

where:

- C_t = Net cash flow at time t,
- r = Discount rate (enterprise rate = **15%**),
- n = Project lifetime (including construction and operation),
- C_0 = Initial capital investment.

A project is considered **economically viable** if:

$$NPV > 0 \tag{88}$$

indicating that the total discounted cash inflows exceed the initial investment.

The internal rate of return (IRR) is the discount rate that makes the NPV equal to zero:

$$\sum_{t=0}^{n} \frac{C_t}{(1+IRR)^t} - C_0 = 0 \tag{89}$$

where:

- IRR is found iteratively by solving for r such that $NPV = 0$.

The project is considered **financially attractive** if:

$$IRR > 15\% \tag{90}$$

indicating that the return rate exceeds the enterprise rate.

8.5.2 Product Sales Revenue

The revenue generated from benzene (B), methane (M), and diphenyl (D) is calculated as:

$$R_{\text{products}} = M_B P_B + M_M P_M + M_D P_D \tag{91}$$

where:

- M_B, M_M, M_D are the production rates of benzene, methane, and diphenyl (tonnes per year).

- P_B, P_M, P_D are their respective market prices (\$/tonne).

8.5.3 Feedstock Cost

The total cost of feedstock (toluene and hydrogen) is:

$$C_{\text{feedstock}} = M_T P_T + M_H P_H \tag{92}$$

where:

- M_T, M_H are the mass flow rates of toluene and hydrogen (tonnes per year).
- P_T, P_H are their respective prices (\$/tonne).

8.5.4 Profit Calculation

The net annual profit is determined as:

$$\text{Profit} = R_{\text{products}} - C_{\text{feedstock}} - C_{\text{utilities}} - C_{\text{maintenance}} \tag{93}$$

8.5.5 Equipment Cost Estimation

The capital cost of major equipment is estimated using the Chemical Engineering Plant Cost Index (CEPCI):

$$C_{\text{equipment}} = C_{\text{base}} \times \left(\frac{\text{CEPCI}_{\text{current}}}{\text{CEPCI}_{\text{base}}} \right) \tag{94}$$

8.5.6 Annual Depreciation

The plant follows a 10-year linear depreciation schedule, calculated as:

$$D_{\text{annual}} = \frac{FCI}{\text{Depreciation Period (10 years)}} \tag{95}$$

Appendix F.HAZOP Analysis

Table 16: HAZOP Analysis of Major Unit Operations.

Item	Parameter	Guide Word	Cause	Consequence	Safeguards & Actions
Reactor	Temperature	High	Excess feed, runaway reaction	Catalyst deactivation, pressure buildup	Temperature alarms, feed ratio control
		Low	Insufficient preheating	Low conversion, byproducts	Heat integration, insulation
Gas Recycle	Hydrogen Flow	More	Malfunction in purge	Energy loss, compressor overloading	Optimize purge, backup compressor
		Less	Recycle blockage	Low conversion, catalyst deactivation	Install alarms, predictive maintenance
Separator	Phase Separation	Poor	Incorrect pressure, turbulence	Low purity, downstream issues	Pressure control, routine maintenance
		High Level	Pump failure	Overfilling, pressure buildup	Automate level control, add drain
Purge Flow	Purge Rate	More	Valve misadjustment	Hydrogen loss, inefficiency	Optimize valve control, gas analysis
		Less	Valve failure	Methane buildup, pressure risk	Valve testing, backup sensors
Compressor	Pressure	More	Discharge blockage	Equipment damage, explosion risk	Relief valves, emergency shutdown
		Less	Leak, failure	Poor reaction efficiency	Pressure alarms, routine maintenance
Heater	Heat Supply	More	Faulty control, sensor error	Overheating, feed degradation	Temperature sensors, auto shutdown
		Less	Low steam, insulation loss	Incomplete reaction, low efficiency	Improve insulation, backup heating

Appendix: G Python Stream Table

Table 17: Streams for HYSYS Simulation of the HDA Process

#	Stream Name	Stream Composition	Molar Flow (gmol/s)
1	Hydrogen feed	95.0mol% H_2, 5.0mol% CH_4	253.691
2	Toluene feed	100.0mol% toluene	48.647
3	Recycle hydrogen	76.1mol% H_2, 23.9mol% CH_4	27.793
4	Recycle toluene	100.0mol% toluene	0.370
5	Combined hydrogen inlet	93.0mol% H_2, 3.0mol% CH_4	281.483
6	Combined toluene inlet	100.0mol% toluene	52.433
7	Toluene inlet heated 1	100.0mol% toluene	52.433
8	Toluene inlet heated 2	100.0mol% toluene	52.433
9	Hydrogen inlet heated 1	93.1mol% H_2, 6.9mol% CH_4	281.483
10	Reactor inlet (before trim)	15.7mol% toluene, 78.5mol% H_2, 5.8mol% CH_4	333.944
11	Reactor inlet	15.7mol% toluene, 78.5mol% H_2, 5.8mol% CH_4	333.944
12	Reactor outlet	1.1mol% toluene, 64.9mol% H_2, 12.6mol% benzene, 20.3mol% CH_4, 1.1mol% diphenyl	333.918
13	Split outlet 1	1.1mol% toluene, 64.9mol% H_2, 12.6mol% benzene, 20.3mol% CH_4, 1.1mol% diphenyl	NA
14	Split outlet 2	1.1mol% toluene, 64.9mol% H_2, 12.6mol% benzene, 20.3mol% CH_4, 1.1mol% diphenyl	NA
15	Reactor outlet (exchanged)	1.1mol% toluene, 64.9mol% H_2, 12.6mol% benzene, 20.3mol% CH_4, 1.1mol% diphenyl	333.918
16	Reactor outlet (cooled)	1.1mol% toluene, 64.9mol% H_2, 12.6mol% benzene, 20.3mol% CH_4, 1.1mol% diphenyl	333.918
17	Flash top	76.1mol% H_2, 23.9mol% CH_4	284.781
18	Flash bottom	7.7mol% toluene, 85.6mol% benzene, 6.7mol% diphenyl	49.136
19	RH1	76.1mol% H_2, 23.9mol% CH_4	284.781
20	TDB1	NA	NA
21	RH2	NA	NA
22	TDB2	7.7mol% toluene, 85.6mol% benzene, 6.7mol% diphenyl	49.136
23	Before purge	76.1mol% H_2, 23.9mol% CH_4	284.781
24	Purge	76.1mol% H_2, 23.9mol% CH_4	256.988
25	Warm recycle hydrogen	76.1mol% H_2, 23.9mol% CH_4	30.493
26	Cooled recycle hydrogen	76.1mol% H_2, 23.9mol% CH_4	30.493
27	TDB	7.7mol% toluene, 85.6mol% benzene, 6.7mol% diphenyl	49.136
28	TDB decompressed	7.7mol% toluene, 85.6mol% benzene, 6.7mol% diphenyl	49.136
29	Distillate 1	100.0mol% benzene	42.060
30	Combined benzene	100.0mol% benzene	42.060
31	Benzene product	100.0mol% benzene	42.060
32	Bottom 1	53.5mol% toluene, 46.5mol% diphenyl	7.076
33	Benzene remnant (Bottom 1)	NA	NA
34	TD	53.5mol% toluene, 46.5mol% diphenyl	7.076
35	Distillate 2	100.0mol% toluene	3.783
36	Bottom 2	100.0mol% diphenyl	3.292
37	Diphenyl, hot	100.0mol% diphenyl	3.292
38	Diphenyl product	100.0mol% diphenyl	3.292
39	RT1	100.0mol% toluene	3.783
40	RT2	100.0mol% toluene	3.783

8.6 Appendix H: Detailed Cash Flow Sheet

Alkyl Products Limited
123 Lagoon Road
Santa Barbara, CA

ECONOMIC ANALYSIS

Gulf Coast Site, Freeport, TX

Case Description: Scaling up benzene prod to 100 kta via HDA process

Project Name								
Rev	Date	BY	APVD	Rev	Date	BY	APVD	
V1.1	2/14/2023	JPC	MFD					
V1.2	45727	YW	SS	CY				

REVENUES AND PRODUCTION COSTS

	$MM/yr
Main product revenue	96.1
Byproduct revenue	12.4
Raw materials cost	67.3
Utilities cost	5.8
Consumables cost	0.0
CO_2 sustainability charge	0.5
VCOP	61.2
Salary and overheads	0.0
Maintenance	0.3
Interest	0.0
AGS	5.7
FCOP	6.0

CAPITAL COSTS

	$MM
ISBL Capital Cost	11.6
OSBL Capital Cost	4.1
Indirect Cost	3.1
Contingency	0.8
Total Fixed Capital Cost	19.6
Working Capital	9.6
Start-up Costs	2.6
Land	0.0
Total Capital Investment	31.8

CONSTRUCTION SCHEDULE

Year	% FC	% WC	% SU	% FCOP	% VCOP
0					
1	33%				
2	33%				
3	33%	100%	100%		
4				100%	100%
5				100%	100%
6+				100%	100%
				100%	100%

ECONOMIC ASSUMPTIONS

On Stream	8000 hr/yr	Discount Rate	15%	Depreciation method	Straight-Line
	333.3333 day/yr	Tax Rate	27%	Depreciation period	10 yrs
Project Life	10 yr	Salvage Value	5%		

CASH FLOW ANALYSIS

All figures in $MM unless indicated

Project Year	Cap. Ex.	Revenue	COM	Gr. Profit	Depron.	Taxable Inc	Taxes Paid	Cash Flow	PV of CF	NPV
0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	-5.0	-5.0	-5.0
1	6.5	0.0	0.0	0.0	0.0	0.0	0.0	-6.5	-5.6	-10.6
2	6.5	0.0	0.0	0.0	0.0	0.0	0.0	-6.5	-4.9	-15.5
3	18.7	0.0	0.0	0.0	0.0	0.0	0.0	-18.7	-12.3	-27.8
4	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	12.4	-15.4
5	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	10.8	-4.7
6	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	9.4	4.7
7	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	8.1	12.8
8	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	7.1	19.9
9	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	6.2	26.0
10	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	5.3	31.4
11	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	4.7	36.0
12	-1.0	96.1	67.2	28.9	2.1	26.7	7.2	22.6	4.2	40.3
13	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	3.5	43.8
14	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	3.1	46.8
15	0.0	96.1	67.2	28.9	2.1	26.7	7.2	21.6	2.7	49.5

ECONOMIC ANALYSIS

NPV	15 years	49.5	$MM	IRR	39%
NPV at yr	12	40.3	$MM		

Figure 11: Detailed Cash Flow Diagram

8.7 Appendix I: Python Code

8.7.1 Mole balance and Reactor Design

```
# Updated PFR Simulation with Temperature-Dependent  H  and Cp (mol/s basis)

import numpy as np
from scipy.integrate import solve_ivp
import pandas as pd

#           1. FORMATION ENTHALPIES (Hf) [J/mol]


Hf = {
    'T':    50170,      # Toluene
    'H':    0,          # H2
    'M':    -74520,     # CH4
    'B':    82930,      # Benzene
    'D':    178.49e03       # Diphenyl (not sure)
}

#           2. HEAT CAPACITY COEFFICIENTS for Cp_i(T) [J/(mol K)]

# From
coeffs = {
    # Hydrogen ( H  )
    'H': (0.27617e05, 0.0956e05, 2.466e03, 0.0376e05, 567.6),

    # Methane ( C H  )
    'M': (0.33298e05, 0.79933e05, 2.0869e03, 0.41602e05, 991.96),

    # Toluene (  C  H  )
    'T': (0.5814e05, 2.863e05, 1.4406e03, 1.898e05, 650.43),

    # Benzene (  C  H  )
    'B': (0.44767e05, 2.3085e05, 1.4792e03, 1.6836e05, 677.66),

    # Diphenyl (  C  H  -  C  H  )     from DIPPR fit
    'D': (1.0759e05, 4.2105e05, 1.9041e03, 4.1785e05, 828.81)
}

R = 8.314  # J/(mol K)
T_ref = 298.15  # Reference temperature [K]

def Cp_i(T, sp):
    """
    Heat capacity Cp of species 'sp' at temperature T [K] using DIPPR 101 form.
    Returns Cp in J/(mol K).
    """
    C1, C2, C3, C4, C5 = coeffs[sp]  # DIPPR coefficients
    term1 = (C3 / T) / np.sinh(C3 / T)
    term2 = (C5 / T) / np.cosh(C5 / T)
    Cp_kmol = C1 + C2 * term1**2 + C4 * term2**2  # J/kmol K
    return Cp_kmol / 1000  # J/mol K

from scipy.integrate import quad

def H_i(T, sp):
    """
    Total molar enthalpy H of species 'sp' at temperature T [K], in J/mol,
    using DIPPR 101 Cp model and numerical integration.
    """
    integrand = lambda T_val: Cp_i(T_val, sp)
    delta_H, _ = quad(integrand, T_ref, T)  # J/mol
    return Hf[sp] + delta_H

def delta_H_rxn1(T):
    """ H1 (T) = H_B + H_M - (H_T + H_H) [J/mol]"""
    #print(f" H1 (T) = {H_i(T,'B') + H_i(T,'M') - H_i(T,'T') + H_i(T,'H')}")
    return H_i(T,'B') + H_i(T,'M') - (H_i(T,'T') - H_i(T,'H'))

def delta_H_rxn2(T):
    """ H2 (T) = H_D + H_H - 2*H_B [J/mol]"""
```

```
    #print(f" H2 (T) = {H_i(T,'D') + H_i(T,'H') - 2*H_i(T,'B')}")
    return H_i(T,'D') + H_i(T,'H') - 2*H_i(T,'B')

#           3. KINETIC PARAMETERS & RATE LAWS


R = 8.314  # J/(mol K)

k1_0_h, Ea1_kcal = 1.188e14, 52.0
k1_0 = k1_0_h/3600       # (L/mol)^0.5/s
Ea1 = Ea1_kcal*4184      # J/mol

k2_0_h, Ea2_kcal = 1.1647e-5, 30.19
k2_0 = k2_0_h/3600       # mol/(L  Pa   s)
Ea2 = Ea2_kcal*4184      # J/mol

def rate_reaction_1(T,C_T,C_H):
    k1 = k1_0 * np.exp(-Ea1/(R*T))
    return k1 * C_T * np.sqrt(C_H)

def rate_reaction_2(T,P,y_B,y_H,y_D):
    k2 = k2_0 * np.exp(-Ea2/(R*T))
    log10Keq = (-734.92/T - 8.1046 + 3.1294*np.log10(T)
                -7.0804e-4*T + 8.523e-8*T**2)
    Keq = 10**log10Keq
    #return k2 * P**2 * ((y_B**2*y_H/Keq) - y_D*y_H)
    return k2 * P**2 * (y_B**2-y_D*y_H/Keq)

#           4. PFR ODE SYSTEM


def pfr_odes(V,y,P):
    # y = [F_T, F_H, F_B, F_M, F_D, F_tot, T]
    F_T,F_H,F_B,F_M,F_D,F_tot,T = y
    Q = F_tot * R * T * 1000.0 / P  # L/s
    C_T,C_H = F_T/Q, F_H/Q
    y_B,y_Hf,y_D = F_B/F_tot, F_H/F_tot, F_D/F_tot
    r1 = rate_reaction_1(T,C_T,C_H)
    r2 = rate_reaction_2(T,P,y_B,y_Hf,y_D)
    dF_T = -r1
    dF_H = -r1 + r2
    dF_B =  r1 - 2*r2
    dF_M =  r1
    dF_D =  r2
    dF_tot = 0.0
    # Dynamic enthalpies
    dH1 = delta_H_rxn1(T)
    dH2 = delta_H_rxn2(T)
    heat = -(dH1*r1 + dH2*r2)  # J/(s L)
    Cp_mix = (F_T*Cp_i(T,'T') + F_H*Cp_i(T,'H') +
              F_B*Cp_i(T,'B') + F_M*Cp_i(T,'M') +
              F_D*Cp_i(T,'D'))
    dT = heat / Cp_mix * 1000.0 / F_tot  # K/s
    #dT = 0.0
    return [dF_T,dF_H,dF_B,dF_M,dF_D,dF_tot,dT]

#           6. ALL STREAM VALUES


def simulate_basis_pfr(T0_C,P_bar,MR,V_span,n_points=100):
    """
    Simulate adiabatic PFR (1 mol/s toluene).
    Inputs:
      T0_C  [ C ], P_bar [bar], MR, V_span [L], n_points
    Outputs dict with:
      'V (L)', 'F_i (mol/s)', 'F_tot (mol/s)', 'T (K)'
    """
    T0 = T0_C + 273.15  # K
    P  = P_bar * 1e5    # Pa
    F_T0 = 1.0          # mol/s
    F_H0 = MR*F_T0
```

```
    F_M0 = (7/95.0)*F_H0
    F_B0 = F_D0 = 0.0
    F_tot0 = F_T0+F_H0+F_M0
    y0 = [F_T0,F_H0,F_B0,F_M0,F_D0,F_tot0,T0]
    V_eval = np.linspace(V_span[0],V_span[1],n_points)
    sol = solve_ivp(lambda V,y: pfr_odes(V,y,P),
                    V_span,y0,t_eval=V_eval,method='BDF')
    F = sol.y[:5]; F_tot=sol.y[5]; T=sol.y[6]
    V = sol.t
    F_T = F[0]  #F_i means reactor outlet molar flow rates
    F_H = F[1]
    F_B = F[2]
    F_M = F[3]
    F_D = F[4]
    F_tot = sol.y[5]
    T = sol.y[6]
    S = F_B / (F_T0 - F_T)
    x = (F_T0-F_T) / F_T0
    q0 = (R*T/P)*F_tot0 * 1000  # L/s
    tau = V/q0
    P_B_desired = 45  # mol/s
        ------------------------------------------------------------------------------
         100 kta benzene
    F_T_R_in_true = P_B_desired / (S * x)    #F_i_true means reactor inlet molar
        flow rates
    F_H_R_in_true = MR * P_B_desired / (S * x)
    F_M_R_in_true = (7/95.0) * F_H_R_in_true
    F_B_R_in_true = np.zeros_like(S)
    F_D_R_in_true = np.zeros_like(S)
    F_tot_R_in_true = F_T_R_in_true + F_H_R_in_true + F_M_R_in_true + F_B_R_in_true
        + F_D_R_in_true
    q0_true = R * T / P * F_tot_R_in_true * 1000  # L/s
    V_true = tau * q0_true

    return {
        'V␣(L)':          V,
        'T␣(K)':          T,
        'F_T␣(mol/s)':    F_T,
        'F_H␣(mol/s)':    F_H,
        'F_B␣(mol/s)':    F_B,
        'F_M␣(mol/s)':    F_M,
        'F_D␣(mol/s)':    F_D,
        'S␣(-)':          S,
        'x␣(-)':          x,
        'tau␣(s)':        tau,
        'F_T_R_in_true␣(mol/s)': F_T_R_in_true,
        'F_H_R_in_true␣(mol/s)': F_H_R_in_true,
        'F_B_R_in_true␣(mol/s)': F_B_R_in_true,
        'F_M_R_in_true␣(mol/s)': F_M_R_in_true,
        'F_D_R_in_true␣(mol/s)': F_D_R_in_true,
        'F_tot_R_in_true␣(mol/s)': F_tot_R_in_true,
        'q0_true␣(L/s)': q0_true,
        'V_true␣(L)': V_true,
        'q0␣(L/s)': q0
    }


# Reactor purchased cost calculation
def reactor_purchased_cost(V_m3,
                           P_bar,
                           CEPCI=800,
                           L_by_D=10,
                           Fm=2.25):
    """
    Estimate the **purchased cost** of an adiabatic PFR.

    Parameters:
      V_m3     : reactor free volume [m ]
      P_bar    : design pressure [bar]
      CEPCI    : Chemical Engineering Plant Cost Index
      L_by_D   : length-to-diameter ratio (typical packed-bed: 8  12  ;
                 higher L/D can improve contact but increases pressure drop)
      Fm       : material factor (from Table D.6; e.g., CS=1.0, SS clad=2.25)
```

```
    Returns:
      C_purch  : purchased equipment cost [$]
    """
    # Convert pressure from bar to psi
    bar_to_psi = 14.5038
    P_psi = P_bar * bar_to_psi

    # 1) Convert volume to ft
    m3_to_ft3 = 35.3147
    V_ft3 = V_m3 * m3_to_ft3

    # 2) Geometry: D and H in ft
    D = (4 * V_ft3 / (3.1416 * L_by_D)) ** (1/3)
    H = L_by_D * D

    # 3) Pressure factor Fp (piecewise from D.9 table)
    if   P_psi <= 50:    Fp = 1.00
    elif P_psi <=100:    Fp = 1.05
    elif P_psi <=200:    Fp = 1.15
    elif P_psi <=300:    Fp = 1.20
    elif P_psi <=400:    Fp = 1.35
    elif P_psi <=500:    Fp = 1.45
    elif P_psi <=600:    Fp = 1.60
    elif P_psi <=700:    Fp = 1.80
    elif P_psi <=800:    Fp = 1.90
    elif P_psi <=900:    Fp = 2.30
    else:                Fp = 2.50

    # 4) Combined correction factor
    Fc = Fm * Fp

    # 5) Purchased cost correlation (D.9)
    C_purch = (CEPCI / 113.7) * (101.9 * D**1.066 * H**0.82 * Fc)
    return C_purch


# Reactor installed cost calculation
def reactor_installed_cost(V_m3,
                           P_bar,
                           CEPCI=800,
                           L_by_D=10,
                           Fm=2.25):
    """
    Estimate the **installed cost (ISBL)** of an adiabatic PFR.

    Parameters:
      V_m3        : reactor free volume [m ]
      P_bar       : design pressure [bar]
      CEPCI       : Chemical Engineering Plant Cost Index
      L_by_D      : length-to-diameter ratio (typical packed-bed: 8  12  ;
                    higher L/D can improve contact but increases pressure drop)
      Fm          : material factor (from Table D.6; e.g., CS=1.0, SS clad=2.25)

    Returns:
      C_installed : installed cost (inside battery limits) [$]
    """
    # Convert pressure from bar to psi
    bar_to_psi = 14.5038
    P_psi = P_bar * bar_to_psi

    # 1) Convert volume to ft
    m3_to_ft3 = 35.3147
    V_ft3 = V_m3 * m3_to_ft3

    # 2) Geometry: D and H in ft
    D = (4 * V_ft3 / (3.1416 * L_by_D)) ** (1/3)
    H = L_by_D * D

    # 3) Pressure factor Fp (same piecewise lookup)
    if   P_psi <= 50:    Fp = 1.00
    elif P_psi <=100:    Fp = 1.05
    elif P_psi <=200:    Fp = 1.15
```

```
    elif P_psi <=300:    Fp = 1.20
    elif P_psi <=400:    Fp = 1.35
    elif P_psi <=500:    Fp = 1.45
    elif P_psi <=600:    Fp = 1.60
    elif P_psi <=700:    Fp = 1.80
    elif P_psi <=800:    Fp = 1.90
    elif P_psi <=900:    Fp = 2.30
    else:                Fp = 2.50

    # 4) Combined correction factor
    Fc = Fm * Fp

    # 5) Installed cost correlation (D.10)
    C_installed = (CEPCI / 113.7) * (101.9 * D**1.066 * H**0.82 * (2.18 + Fc))
    return C_installed

def furnace_cost(
    Q_W,                    # required furnace duty [W]
    design_type,            # 'heater', 'pyrolysis', or 'reformer'
    tube_material,          # 'cs', 'chromemoly', or 'stainless'
    P_bar,                  # design pressure [bar]
    CEPCI=800.0             # cost index
):
    """
    Returns (C_purchased, C_installed) [$] for a process furnace.

    Correlations (D.1 & D.2):
      C_purch = (CEPCI/113.7) * (5.52 10 ^3) * Q^0.85 * Fc
      C_inst  = (CEPCI/113.7) * (5.52 10 ^3) * Q^0.85 * (1.27 + Fc)

    where Q is in 10^6 Btu/hr, and
      Fc = Fd + Fm + Fp

    Correction factors (Table D.1):
      design_type      Fd:
        'heater'   : 1.00
        'pyrolysis': 1.10
        'reformer' : 1.35

      tube_material      Fm:
        'cs'         : 0.00   # carbon steel
        'chromemoly' : 0.35
        'stainless'  : 0.75

      pressure (psi)      Fp:
            500     : 0.00
           1000     : 0.10
           1500     : 0.15
           2000     : 0.25
           2500     : 0.40
         >2500      : 0.60
    """
    # 1) Convert duty [W]      [Btu/hr]
    Btu_per_hr = Q_W * 3.412  # 1 W = 3.412 Btu/hr
    Q_million = Btu_per_hr / 1e6

    # 2) Select Fd
    fd_map = {
        'heater':    1.00,
        'pyrolysis': 1.10,
        'reformer':  1.35
    }
    try:
        Fd = fd_map[design_type.lower()]
    except KeyError:
        raise ValueError(f"design_type must be one of {list(fd_map)}")

    # 3) Select Fm
    fm_map = {
        'cs':          0.00,
        'chromemoly':  0.35,
        'stainless':   0.75
    }
```

```
    try:
        Fm = fm_map[tube_material.lower()]
    except KeyError:
        raise ValueError(f"tube_material must be one of {list(fm_map)}")

    # 4) Pressure factor Fp (psi)
    psi = P_bar * 14.5038
    if   psi <=  500: Fp = 0.00
    elif psi <= 1000: Fp = 0.10
    elif psi <= 1500: Fp = 0.15
    elif psi <= 2000: Fp = 0.25
    elif psi <= 2500: Fp = 0.40
    else:             Fp = 0.60

    # 5) Combined Fc
    Fc = Fd + Fm + Fp

    # 6) Base multiplier
    factor = (CEPCI / 113.7) * 5.52e3

    # 7) Cost correlations
    C_purchased = factor * (Q_million ** 0.85) * Fc
    C_installed = factor * (Q_million ** 0.85) * (1.27 + Fc)

    return C_purchased, C_installed


def compressor_cost(
    power_W,             # required compression power [W]
    machine_type,        # one of: 'centrifugal, motor'
                         #         'centrifugal, turbine'
                         #         'reciprocating, steam'
                         #         'reciprocating, motor'
                         #         'reciprocating, gas engine'
    efficiency=0.90,     # mechanical efficiency (bhp = thp/eff)
    CEPCI=800.0          # cost index
):
    """
    Returns (C_purchased, C_installed) [$] for a gas compressor.

    Correlations (D.7 & D.8):
      C_purch  = (CEPCI/113.7) * [517.5 * (bhp)^0.82] * Fc
      C_inst   = (CEPCI/113.7) * [517.5 * (bhp)^0.82] * (2.11 + Fc)

    where:
      bhp =  b r a k e horsepower  = (power_W / 745.7 W/hp) / efficiency
      Fc  = Fd ( d e s i g n type  factor) from Table D.5
    """
    # 1) Map  d e s i g n type      Fd
    fd_map = {
        'centrifugal, motor':       1.00,
        'centrifugal, turbine':     1.15,
        'reciprocating, steam':     1.07,
        'reciprocating, motor':     1.29,
        'reciprocating, gas engine':1.82
    }
    key = machine_type.lower()
    if key not in fd_map:
        raise ValueError(f"machine_type must be one of {list(fd_map)}")
    Fc = fd_map[key]

    # 2) Brake horsepower
    hp_theoretical = power_W / 745.7
    bhp = hp_theoretical / efficiency

    # 3) Base multiplier
    factor = CEPCI / 113.7
    base   = 517.5 * (bhp ** 0.82)

    # 4) Purchase & installed costs
    C_purchased  = factor * base * Fc
    C_installed  = factor * base * (2.11 + Fc)
```

```
    return C_purchased, C_installed

def operating_cost(
    F_T_fresh, F_H2_fresh, F_CH4_fresh,
    F_B_out,   F_D_out,   F_H2_out,  F_CH4_out,
    years=12, hours_per_year=8000
):

    # --- 1) CORRECT UNIT CONVERSIONS ($/mol) ---
    B_price_USD_ton   = 1000.0
    T_price_USD_ton   = 300.0
    gas_price_USD_ton = 1400.0  # blend

    MW_B = 78.11   # g/mol
    MW_T = 92.14
    # benzene: 1000 $/ton     1000/1e6 $/g     *78.11 g/mol
    B_price_USD_mol   = B_price_USD_ton   / 1e6 * MW_B
    T_price_USD_mol   = T_price_USD_ton   / 1e6 * MW_T
    # gas blend: assume 1 mol of blend weighs ~1.1 g (0.95*2 + 0.05*16 = 2.7 g?
    # actually calculate precisely if needed; here  w e ll  assume 2.7 g/mol.)
    MW_gas_blend = 0.95*2.016 + 0.05*16.04  #     2.30 g/mol
    gas_price_USD_mol = gas_price_USD_ton / 1e6 * MW_gas_blend

    fuel_value_USD_GJ   = 4.25   # $ per GJ
    CO2_charge_USD_ton  = 40.0
    MW_CO2 = 44.01  # g/mol
    CO2_charge_USD_mol = CO2_charge_USD_ton / 1e6 * MW_CO2

    # --- 2) Combustion energies [kJ/mol] ---
    D_comb_E = 6299.9
    H_comb_E = 285.8
    M_comb_E = 890.8

    # --- 3) Seconds per year ---
    sec_per_year = hours_per_year * 3600

    # --- 4) Annual mol flows [mol/yr] ---
    T_year    = F_T_fresh   * sec_per_year
    gas_year  = (F_H2_fresh + F_CH4_fresh) * sec_per_year
    B_year    = F_B_out     * sec_per_year

    # --- 5) Feed costs [$] ---
    cost_T_year   = T_year   * T_price_USD_mol
    cost_gas_year = gas_year * gas_price_USD_mol
    feed_cost_yearly   = cost_T_year + cost_gas_year
    feed_cost_lifetime = feed_cost_yearly * years

    # --- 6) Benzene revenues [$] ---
    revenue_B_yearly   = B_year * B_price_USD_mol
    revenue_B_lifetime = revenue_B_yearly * years

    # --- 7) Fuel credit [$] ---
    energy_rate_kJ_s     = (F_D_out*D_comb_E + F_H2_out*H_comb_E + F_CH4_out*
        M_comb_E)
    energy_GJ_per_year   = energy_rate_kJ_s * sec_per_year / 1e6
    fuel_credit_yearly   = energy_GJ_per_year * fuel_value_USD_GJ
    fuel_credit_lifetime = fuel_credit_yearly * years

    # --- 8)  C O   charges [$] ---
    CO2_rate             = 12*F_D_out + 1*F_CH4_out
    CO2_year             = CO2_rate * sec_per_year
    CO2_charge_yearly    = CO2_year * CO2_charge_USD_mol
    CO2_charge_lifetime  = CO2_charge_yearly * years

    # --- 9) Net operating cost ---
    net_op_yearly    = - feed_cost_yearly + revenue_B_yearly + fuel_credit_yearly -
        CO2_charge_yearly
    net_op_lifetime  = net_op_yearly * years

    return {
      'feed_cost_yearly␣($)':       feed_cost_yearly,
      'feed_cost_lifetime␣($)':     feed_cost_lifetime,
      'benzene_revenue_yearly␣($)':  revenue_B_yearly,
```

```
        'benzene_revenue_lifetime ($)':revenue_B_lifetime,
        'fuel_credit_yearly ($)':      fuel_credit_yearly,
        'fuel_credit_lifetime ($)':    fuel_credit_lifetime,
        'CO2_charge_yearly ($)':       CO2_charge_yearly,
        'CO2_charge_lifetime ($)':     CO2_charge_lifetime,
        'net_op_cost_yearly ($)':      net_op_yearly,
        'net_op_cost_lifetime ($)':    net_op_lifetime
    }

def separation_cost(df):
    """
    Calculates separation OPEX, CAPEX, and total separation FCI.
    Assumes black-box separation, 25 C ,   =50, C$/W=0.6.

    Input:
    - df: DataFrame containing reactor outlet flows and purge fraction

    Returns:
    - dict with separation OPEX, separation CAPEX, separation FCI
    """

    # Constants
    R = 8.314  # J/mol/K
    T_sep = 298.15  # K
    lambda_factor = 50
    energy_cost_per_GJ = 4.25  # $/GJ
    C_dollar_per_Watt = 0.6  # $/W
    sec_per_year = 30.24e6
    factor_FCI = 2.5

    # Extract streams
    F_T_out = df['Toluene Reactor Outlet (mol/s)'].values
    F_H2_out = df['Hydrogen Reactor Outlet (mol/s)'].values
    F_CH4_out = df['Methane Reactor Outlet (mol/s)'].values
    F_B_out = df['Benzene Reactor Outlet (mol/s)'].values
    F_D_out = df['Diphenyl Reactor Outlet (mol/s)'].values

    # Separator 1 (Gas vs Liquid)
    F_total_sep1 = F_T_out + F_H2_out + F_CH4_out + F_B_out + F_D_out
    z_H2_sep1 = F_H2_out / (F_total_sep1 + 1e-8)
    z_CH4_sep1 = F_CH4_out / (F_total_sep1 + 1e-8)
    z_T_sep1 = F_T_out / (F_total_sep1 + 1e-8)
    z_B_sep1 = F_B_out / (F_total_sep1 + 1e-8)
    z_D_sep1 = F_D_out / (F_total_sep1 + 1e-8)

    Wmin_sep1 = (F_H2_out*R*T_sep*np.log(1/(z_H2_sep1+1e-8)) +
                 F_CH4_out*R*T_sep*np.log(1/(z_CH4_sep1+1e-8)) +
                 F_T_out*R*T_sep*np.log(1/(z_T_sep1+1e-8)) +
                 F_B_out*R*T_sep*np.log(1/(z_B_sep1+1e-8)) +
                 F_D_out*R*T_sep*np.log(1/(z_D_sep1+1e-8)))

    # Separator 2 (Diphenyl separation)
    F_bottom_sep1 = F_T_out + F_B_out + F_D_out
    z_T_sep2 = F_T_out / (F_bottom_sep1 + 1e-8)
    z_B_sep2 = F_B_out / (F_bottom_sep1 + 1e-8)
    z_D_sep2 = F_D_out / (F_bottom_sep1 + 1e-8)

    Wmin_sep2 = (F_T_out*R*T_sep*np.log(1/(z_T_sep2+1e-8)) +
                 F_B_out*R*T_sep*np.log(1/(z_B_sep2+1e-8)) +
                 F_D_out*R*T_sep*np.log(1/(z_D_sep2+1e-8)))

    # Separator 3 (Benzene/Toluene splitter)
    F_top_sep2 = F_T_out + F_B_out
    z_T_sep3 = F_T_out / (F_top_sep2 + 1e-8)
    z_B_sep3 = F_B_out / (F_top_sep2 + 1e-8)

    Wmin_sep3 = (F_T_out*R*T_sep*np.log(1/(z_T_sep3+1e-8)) +
                 F_B_out*R*T_sep*np.log(1/(z_B_sep3+1e-8)))

    # Total minimum work
    Wmin_total = Wmin_sep1 + Wmin_sep2 + Wmin_sep3

    # Real work
```

```
    Wreal_total = lambda_factor * Wmin_total

    # Separation OPEX ($/yr)
    separation_OPEX = energy_cost_per_GJ * Wreal_total * sec_per_year * 1e-9

    # Separation CAPEX (ISBL)
    separation_CAPEX = C_dollar_per_Watt * Wreal_total

    # Total FCI
    separation_FCI = factor_FCI * separation_CAPEX

    return {
        'Separation OPEX ($/yr)': separation_OPEX,
        'Separation CAPEX ($)': separation_CAPEX,
        'Separation FCI ($)': separation_FCI
    }


def compute_true_outlet_profiles(res, P_B_desired, MR, T0_C, P_bar):
    """
    From basis simulation result `res`, compute true-inlet     outlet profiles.

    Inputs:
      res              : dict from simulate_basis_pfr, containing:
                           'V (L)', 'S (-)', 'x (-)', 'F_tot (mol/s)'
      P_B_desired      : desired benzene purge rate [mol/s]
      MR               : H2-to-T molar ratio
      T0_C             : inlet temperature [ C ]
      P_bar            : pressure [bar]

    Returns a dict of arrays (length = len(res['V (L)'])):
      'V (L)'                : basis volumes [L]
      'tau (s)'              : basis residence times [s]
      'F_T_R_true (mol/s)'   : true toluene inlet
      'F_H_R_true (mol/s)'   : true hydrogen inlet
      'F_M_R_true (mol/s)'   : true methane inlet
      'F_tot_R_true (mol/s)': true total inlet
      'F_T_out (mol/s)'      : outlet toluene
      'F_H_out (mol/s)'      : outlet hydrogen
      'F_B_out (mol/s)'      : outlet benzene
      'F_M_out (mol/s)'      : outlet methane
      'F_D_out (mol/s)'      : outlet diphenyl
      'T_out (K)'            : outlet temperature
    """
    import numpy as np
    from scipy.integrate import solve_ivp

    # Extract basis results
    V_basis     = res['V_true (L)']
    S           = res['S (-)']
    x           = res['x (-)']
    F_T_R_true = res['F_T_R_in_true (mol/s)']
    F_H_R_true = res['F_H_R_in_true (mol/s)']
    F_B_R_true = res['F_B_R_in_true (mol/s)']
    F_M_R_true = res['F_M_R_in_true (mol/s)']
    F_D_R_true = res['F_D_R_in_true (mol/s)']
    F_tot_R_true = res['F_tot_R_in_true (mol/s)']


    # Reactor conditions
    T0 = T0_C + 273.15       # K
    P0 = P_bar * 1e5         # Pa

    # Prepare output arrays
    n = len(V_basis)
    F_T_out = np.zeros(n)
    F_H_out = np.zeros(n)
    F_B_out = np.zeros(n)
    F_M_out = np.zeros(n)
    F_D_out = np.zeros(n)
    T_out   = np.zeros(n)
    rpc = np.zeros(n)
    ric = np.zeros(n)
```

```
    # Integrate PFR for each basis volume
    for i in range(1,n):
        y0 = [
            F_T_R_true[i],
            F_H_R_true[i],
            F_B_R_true[i],
            F_M_R_true[i],
            F_D_R_true[i],
            F_tot_R_true[i],
            T0
        ]
        V_end = V_basis[i]
        sol = solve_ivp(
            lambda V, y: pfr_odes(V, y, P0),
            [0.0, V_end],
            y0,
            t_eval=[V_end],
            method='BDF'
        )
        F_T_out[i], F_H_out[i], F_B_out[i], F_M_out[i], F_D_out[i], _, T_out[i] =
            sol.y[:, -1]

        rpc[i] = reactor_purchased_cost(V_basis[i] / 1000, P_bar)  # Convert L to
            m
        ric[i] = reactor_installed_cost(V_basis[i] / 1000, P_bar)  # Convert L to
            m

        # 25C reactor inlet H
        H_reactor_in_25C = (F_T_R_true[i] * Cp_i(298, 'T') * 298 +
                 F_H_R_true[i] * Cp_i(298, 'H') * 298 +
                 F_B_R_true[i] * Cp_i(298, 'B') * 298 +
                 F_M_R_true[i] * Cp_i(298, 'M') * 298 +
                 F_D_R_true[i] * Cp_i(298, 'D') * 298)
        # 620C reactor inlet H
        H_reactor_in_620C = (F_T_out[i] * Cp_i(620, 'T') * 893 +
                 F_H_out[i] * Cp_i(620, 'H') * 893 +
                 F_B_out[i] * Cp_i(620, 'B') * 893 +
                 F_M_out[i] * Cp_i(620, 'M') * 893 +
                 F_D_out[i] * Cp_i(620, 'D') * 893)
        # reactor outlet H
        H_reactor_out = (F_T_out[i] * Cp_i(T_out[i], 'T') * T_out[i] +
                 F_H_out[i] * Cp_i(T_out[i], 'H') * T_out[i] +
                 F_B_out[i] * Cp_i(T_out[i], 'B') * T_out[i] +
                 F_M_out[i] * Cp_i(T_out[i], 'M') * T_out[i] +
                 F_D_out[i] * Cp_i(T_out[i], 'D') * T_out[i])




    return {
        'V (L)':                 V_basis,
        'F_T_R_true (mol/s)':    F_T_R_true,
        'F_H_R_true (mol/s)':    F_H_R_true,
        'F_M_R_true (mol/s)':    F_M_R_true,
        'F_tot_R_true (mol/s)': F_tot_R_true,
        'F_T_out (mol/s)':       F_T_out,
        'F_H_out (mol/s)':       F_H_out,
        'F_B_out (mol/s)':       F_B_out,
        'F_M_out (mol/s)':       F_M_out,
        'F_D_out (mol/s)':       F_D_out,
        'T_out (K)':             T_out,
        'Reactor Purchased Cost ($)': rpc,
        'Reactor Installed Cost ($)': ric
    }


# Example usage
if __name__ == "__main__":
    res = simulate_basis_pfr(580,34,MR=5.0,V_span=(0,20000),n_points=500)    #
        variable "res" used in the function compute_true_outlet_profiles, so don't
```

```
        change name without changing the name in the function!
    true_res = compute_true_outlet_profiles(res, P_B_desired=45, MR=5.0,T0_C=580,
        P_bar=34.0)
    #                 Purge and Recycle Mass Balance

    def calculate_precise_purge_and_fresh_feeds(F_H2_out, F_CH4_out, F_Toluene_inlet
        , MR=5.0):
        """
        Vectorized calculation of precise purge fraction and fresh hydrogen feed.

        Inputs:
        - F_H2_out: hydrogen outlet molar flow (array)
        - F_CH4_out: methane outlet molar flow (array)
        - F_Toluene_inlet: toluene inlet molar flow (array)
        - MR: hydrogen to toluene molar ratio (default 5.0)

        Returns:
        - purge_fraction (array)
        - F_H2_fresh (array)
        - F_CH4_fresh (array)
        """

        epsilon = 1e-8  # to prevent division by zero

        F_M_gen = (F_CH4_out - (F_Toluene_inlet * 5 * 7 / 95))

        # Equation A: Solve fresh H2
        F_H2_fresh = (5*F_Toluene_inlet - F_H2_out*(1-F_M_gen/F_CH4_out))/(1-5/95 *
            (F_H2_out/(F_CH4_out + epsilon)))

        # Fresh CH4 from fresh H2
        F_CH4_fresh = (0.05/0.95) * F_H2_fresh

        # Equation B: Solve purge fraction
        purge_fraction = (F_CH4_fresh + F_M_gen) / (F_CH4_out + epsilon)



        return purge_fraction, F_H2_fresh, F_CH4_fresh



    # Reactor outlet H2 and CH4
    F_H2_out = true_res['F_H_out␣(mol/s)']
    F_CH4_out = true_res['F_M_out␣(mol/s)']
    F_Toluene_inlet = res['F_T_R_in_true␣(mol/s)']


# Solve
purge_fraction, F_H2_fresh, F_CH4_fresh = calculate_precise_purge_and_fresh_feeds(
    F_H2_out,
    F_CH4_out,
    F_Toluene_inlet,
)

# Solve for fresh Toluene feed
F_Toluene_inlet = res['F_T_R_in_true␣(mol/s)']
F_Toluene_outlet = true_res['F_T_out␣(mol/s)']
F_T_fresh = F_Toluene_inlet - F_Toluene_outlet

# Solve for Recycle flow rates
F_T_Rec = (1 - purge_fraction) * F_Toluene_outlet
F_H_Rec = (1 - purge_fraction) * F_H2_out
F_M_Rec = (1 - purge_fraction) * F_CH4_out

# Solve for Purge flow rates
F_T_Purge = purge_fraction * F_Toluene_outlet
F_H_Purge = purge_fraction * F_H2_out
F_M_Purge = purge_fraction * F_CH4_out


# Solve for operating cost
operating_cost_result = operating_cost(
```

```
    F_T_fresh,
    F_H2_fresh,
    F_CH4_fresh,
    true_res['F_B_out (mol/s)'],
    true_res['F_D_out (mol/s)'],
    F_H_Purge,
    F_M_Purge
)
total_operating_cost = operating_cost_result['net_op_cost_lifetime ($)']
feed_cost = operating_cost_result['feed_cost_lifetime ($)']
benzene_revenue = operating_cost_result['benzene_revenue_lifetime ($)']
fuel_credit = operating_cost_result['fuel_credit_lifetime ($)']
CO2_charge = operating_cost_result['CO2_charge_lifetime ($)']

# Total cost
total_cost = total_operating_cost - true_res['Reactor Purchased Cost ($)'] -
    true_res['Reactor Installed Cost ($)']

# Collect your outputs into a DataFrame
df = pd.DataFrame({
    'V (L)': res['V (L)'],
    'V_true (L)': res['V_true (L)'],
    'T (K)': true_res['T_out (K)'],
    'S (-)': res['S (-)'],
    'x (-)': res['x (-)'],
    'Toluene Fresh Feed (mol/s)': F_T_fresh,
    'Hydrogen Fresh Feed (mol/s)': F_H2_fresh,
    'Methane Fresh Feed (mol/s)': F_CH4_fresh,
    'Toulene Reactor Inlet (mol/s)': res['F_T_R_in_true (mol/s)'],
    'Hydrogen Reactor Inlet (mol/s)': res['F_H_R_in_true (mol/s)'],
    'Methane Reactor Inlet (mol/s)': res['F_M_R_in_true (mol/s)'],
    'Volumetric Reactor Inlet (L/s)': res['q0_true (L/s)'],
    'Toluene Reactor Outlet (mol/s)': true_res['F_T_out (mol/s)'],
    'Hydrogen Reactor Outlet (mol/s)': true_res['F_H_out (mol/s)'],
    'Methane Reactor Outlet (mol/s)': true_res['F_M_out (mol/s)'],
    'Benzene Reactor Outlet (mol/s)': true_res['F_B_out (mol/s)'],
    'Diphenyl Reactor Outlet (mol/s)': true_res['F_D_out (mol/s)'],
    'T_out (K)': true_res['T_out (K)'],
    'Purge Fraction (-)': purge_fraction,
    'Toluene Recycle (mol/s)': F_T_Rec,
    'Hydrogen Recycle (mol/s)': F_H_Rec,
    'Methane Recycle (mol/s)': F_M_Rec,
    'Toluene Purge (mol/s)': F_T_Purge,
    'Hydrogen Purge (mol/s)': F_H_Purge,
    'Methane Purge (mol/s)': F_M_Purge,
    'Reactor Purchased Cost ($)': true_res['Reactor Purchased Cost ($)'],
    'Reactor Installed Cost ($)': true_res['Reactor Installed Cost ($)'],
    'Operating Cost ($)': total_operating_cost,
    'Feed Cost ($)': feed_cost,
    'Benzene Revenue ($)': benzene_revenue,
    'Fuel Credit ($)': fuel_credit,
    'CO2 Charge ($)': CO2_charge,
    'Total Cost ($)': total_cost
})

#           ADD SEPARATION COSTS



# Calculate separation costs using your function
separation_result = separation_cost(df)

# Unpack results
separation_OPEX = separation_result['Separation OPEX ($/yr)']
separation_CAPEX = separation_result['Separation CAPEX ($)']
separation_FCI = separation_result['Separation FCI ($)']

# Add separation costs to DataFrame
df['Separation OPEX ($/yr)'] = separation_OPEX
df['Separation CAPEX ($)'] = separation_CAPEX
df['Separation FCI ($)'] = separation_FCI
```

```
#            UPDATE PLANT ECONOMICS


# Compute Updated Net Operating Cost (OPEX including separation)
df['Updated OPEX ($/yr)'] = df['Operating Cost ($)'] / 12 + df['Separation OPEX ($/
    yr)']

# Compute Updated Total Capital Investment (TCI)
df['Total TCI ($)'] = df['Reactor Installed Cost ($)'] + df['Separation FCI ($)']

# Compute 12-Year Total Revenue (Benzene Revenue + Fuel Credit - Feed Cost - CO2
    Charge)    12
df['12yr Total Revenue ($)'] = (df['Benzene Revenue ($)']/12 + df['Fuel Credit ($)'
    ]/12
                                - df['Feed Cost ($)']/12 - df['CO2 Charge ($)']/12)
                                    * 12

# Compute 12-Year Plant Profit
df['Plant Profit (12yr) ($)'] = df['12yr Total Revenue ($)'] - df['Total TCI ($)']

#            DISPLAY FULL DATAFRAME


pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', '{:,.3f}'.format)
display(df)
```

8.7.2 Ternary Distillation System USing Underwood

```
import numpy as np
from scipy.optimize import fsolve
import math


# Step 1: Problem Setup
F = 44.8056  # Feed flowrate (mol/s)
z = {'A': 0.0893, 'B': 0.8751, 'C': 0.0356}  # Feed composition
alpha = {
    'A': 27.1543,
    'B':53.6691,
    'C': 1.0  # Reference component (Diphenyl)
}
alpha_vals = sorted(alpha.values())
print("check")
print(alpha_vals)
# Step 2: Solve the mole balance equations
# Unknowns: D (distillate), B (bottom), xC_B (C in bottoms), xB_D (B in distillate)
# Step 1: Feed and composition setup

# Step 2: Distillate flowrate (all benzene goes to D)
D = F * z['B']
B = F - D

# Step 3: Distillate composition specification
x_D = {'A': 0.001, 'B': 0.999, 'C': 0.0}
xA_D = 0.001
xB_D = 0.999
# Step 4: Compute bottom composition by component balance
x_B = {
    'A': (F * z['A']) / B,
    'B': 0.0,  # all B goes to distillate
    'C': (F * z['C']) / B
}

# Step 5: Print updated mole balances
print("   Updated Mole Balances:")
print(f"Feed Flowrate:      F = {F:.4f} mol/s")
print(f"Distillate Flowrate: D = {D:.4f} mol/s")
print(f"Bottom Flowrate:    B = {B:.4f} mol/s")

print("\n--- Distillate Composition (x_D) ---")
for k, v in x_D.items():
    print(f"{k}: {v:.4f}")

print("\n--- Bottom Composition (x_B) ---")
for k, v in x_B.items():
    print(f"{k}: {v:.4f}")

# Step 4: Define Underwood equation
def underwood_eq(theta):
    return sum((alpha[i] * z[i]) / (alpha[i] - theta) for i in alpha)

# Bisection method for root finding
def bisection(f, a, b, tol=1e-10, max_iter=1000):
    fa, fb = f(a), f(b)
    if fa * fb > 0:
        return None
    for _ in range(max_iter):
        c = (a + b) / 2
        fc = f(c)
        if abs(fc) < tol or (b - a) / 2 < tol:
            return c
        if fa * fc < 0:
            b, fb = c, fc
        else:
            a, fa = c, fc
    return None

# Step 5: Find Underwood root
theta1 = None
for i in range(len(alpha_vals) - 1):
```

```
    a = alpha_vals[i] + 1e-4
    b = alpha_vals[i + 1] - 1e-4
    root = bisection(underwood_eq, a, b)
    if root is not None:
        theta1 = root
        break

# Step 6: Calculate r_min using simplified 2-component formula
rmin = (alpha['A'] * xA_D) / (alpha['A'] - theta1) + (alpha['B'] * xB_D) / (alpha['B
    '] - theta1) - 1

# Step 7: Operational Boil-Up Ratio
r = 2 * rmin
q = 1.0
D_to_B = D / B
s = D_to_B * (r + q) - (1 - q)
smin = D_to_B*(rmin +q) - (1- q)
print(smin)
# Step 8: Print results
print("=== Mole Balance & Separation Results ===")
print(f"Feed Flowrate (F): {F:.4f} mol/s")
print(f"Distillate Flowrate (D): {D:.4f} mol/s")
print(f"Bottom Flowrate (B): {B:.4f} mol/s")

print("\n--- Distillate Composition (x_D) ---")
for k, v in x_D.items():
    print(f"{k}: {v:.4f}")

print("\n--- Bottom Composition (x_B) ---")
for k, v in x_B.items():
    print(f"{k}: {v:.4f}")

print("\n--- Underwood & Reflux ---")
print(f"Underwood Root      = {theta1:.6f}")
print(f"Minimum Reflux Ratio r_min = {rmin:.6f}")

print("\n--- Operational Boil-Up Ratio ---")
print(f"Boil-Up Ratio (s) = {s:.6f}")

x_f_NT = {
    'A': 0.000001,     # Toluene
    'B': 0.999,     # Benzene
    'C': 0.000999      # Diphenyl
}

# Assume D is known (for example, recover almost all benzene in D)
D = F * z['B']  # Assume almost all benzene goes to distillate

# Now B
B = F - D

x_f = {}
for k in alpha:
    x_f[k] = (q * z[k] + (1 - q) * x_D[k])

print("--- Feed Stage Composition (x_f) ---")
for k, v in x_f.items():
    print(f"{k}: {v:.4f}")

# Now calculate bottom composition x_B_new
x_B_new = {}
for i in z:
    x_B_new[i] = (F * z[i] - D * x_f_NT[i]) / B

print(x_B_new)
# --- Step 8: Find roots ---
def rectifying_underwood_eq(phi):
    try:
        return sum((alpha[i] * x_f_NT[i]) / (alpha[i] - phi) for i in alpha) - (rmin
            + 1)
    except ZeroDivisionError:
        return float('inf')
```

```
def stripping_underwood_eq(phi):
    try:
        return s + sum((alpha[i] * x_B_new[i]) / (alpha[i] - phi) for i in alpha)
    except ZeroDivisionError:
        return float('inf')

# Bisection root finder

def bisection(f, a, b, tol=1e-10, max_iter=1000):
    fa, fb = f(a), f(b)
    if fa * fb > 0:
        return None
    for _ in range(max_iter):
        c = (a + b) / 2
        fc = f(c)
        if abs(fc) < tol or (b - a) / 2 < tol:
            return c
        if fa * fc < 0:
            b, fb = c, fc
        else:
            a, fa = c, fc
    return None

def find_multiple_roots_safe(f, alpha_vals, tol=1e-10):
    roots = []
    offset = 1e-3
    sorted_alpha = sorted(alpha_vals)
    intervals = []
    intervals.append((min(sorted_alpha) - 2, min(sorted_alpha) - offset))
    for i in range(len(sorted_alpha) - 1):
        intervals.append((sorted_alpha[i] + offset, sorted_alpha[i+1] - offset))
    intervals.append((max(sorted_alpha) + offset, max(sorted_alpha) + 2))
    for (a, b) in intervals:
        test_points = np.linspace(a, b, 1000)
        for i in range(len(test_points) - 1):
            x0, x1 = test_points[i], test_points[i + 1]
            try:
                if f(x0) * f(x1) < 0:
                    root = bisection(f, x0, x1, tol)
                    if root is not None and all(abs(root - existing) > 1e-6 for
                        existing in roots):
                        roots.append(root)
            except:
                continue
    return sorted(roots)

# Find roots
rectifying_roots = find_multiple_roots_safe(rectifying_underwood_eq, alpha_vals)
stripping_roots = find_multiple_roots_safe(stripping_underwood_eq, alpha_vals)

print("\n--- Rectifying Section Roots ---")
for i, root in enumerate(rectifying_roots):
    print(f"Rectifying Root {i+1}:  = {root:.6f}")

print("\n--- Stripping Section Roots ---")
for i, root in enumerate(stripping_roots):
    print(f"Stripping Root {i+1}:  = {root:.6f}")

# NT and NB calculations

def calculate_NT(alpha, x_f, phi_j, phi_k):
    numerator = sum((alpha[i] * x_f[i]) / (alpha[i] - phi_j) for i in alpha)
    denominator = sum((alpha[i] * x_f[i]) / (alpha[i] - phi_k) for i in alpha)
    if numerator/denominator > 0 and phi_j > phi_k:
        return round(math.log(numerator / denominator) / math.log(phi_k / phi_j))
    else:
        return None

def calculate_NB(alpha, x_f, phi_j, phi_k):
    numerator = sum((alpha[i] * x_f[i]) / (alpha[i] - phi_k) for i in alpha)
    denominator = sum((alpha[i] * x_f[i]) / (alpha[i] - phi_j) for i in alpha)
    if numerator/denominator > 0 and phi_j > phi_k:
        return round(math.log(numerator / denominator) / math.log(phi_k / phi_j))
```

```
    else:
        return None

# Calculate NT
if len(rectifying_roots) >= 2:
    if len(rectifying_roots) == 2:
        phi_j, phi_k = sorted(rectifying_roots, reverse=True)
    else:
        valid_rectifying_roots = sorted(
            [r for r in rectifying_roots if min(alpha.values()) < r < max(alpha.
                values())],
            reverse=True
        )
        if len(valid_rectifying_roots) >= 2:
            phi_j, phi_k = valid_rectifying_roots[0], valid_rectifying_roots[1]
        else:
            phi_j = phi_k = None

    if phi_j and phi_k:
        NT = calculate_NT(alpha, x_f, phi_j, phi_k)
        if NT is not None:
            print(f"\ n  ␣Number␣of␣Rectifying␣Stages␣(NT)␣=␣{NT}␣using␣     ␣=␣{
                phi_j:.6f},␣     ␣=␣{phi_k:.6f}")
        else:
            print("\ n  ␣NT␣could␣not␣be␣computed.")
    else:
        print("\ n  ␣Not␣enough␣valid␣rectifying␣roots.")
else:
    print("\ n  ␣NT␣roots␣could␣not␣be␣found.")

# --- Calculate NB ---
def compute_valid_nb_from_roots(stripping_roots, alpha, x_f):
    results = []
    for i in range(len(stripping_roots)):
        for j in range(len(stripping_roots)):
            if i == j:
                continue
            phi_j = stripping_roots[i]
            phi_k = stripping_roots[j]
            if phi_j > phi_k:
                numerator = sum((alpha[n] * x_f[n]) / (alpha[n] - phi_k) for n in
                    alpha)
                denominator = sum((alpha[n] * x_f[n]) / (alpha[n] - phi_j) for n in
                    alpha)
                try:
                    ratio = numerator / denominator
                    if ratio > 0:
                        NB = round(math.log(ratio) / math.log(phi_k / phi_j), 4)
                        results.append({
                            "phi_j": phi_j,
                            "phi_k": phi_k,
                            "numerator": numerator,
                            "denominator": denominator,
                            "NB": NB
                        })
                except (ZeroDivisionError, ValueError):
                    continue

    if results:
        for res in results:
            print(f"   ␣NB␣=␣{res['NB']}␣using␣     ␣=␣{res['phi_j']:.6f},␣     ␣=␣{
                res['phi_k']:.6f}")
    else:
        print("   ␣No␣valid␣NB␣found␣with␣positive␣ratio.")

    return results

nb_results = compute_valid_nb_from_roots(stripping_roots, alpha, x_f)
if nb_results:
    NB = nb_results[0]['NB']

if 'NT' in locals() and 'NB' in locals() and NT is not None and NB is not None:
    Neq = math.ceil(NT + NB+1)
```

```
    Nreal = math.ceil(Neq / 0.3)  # Always round up
    print("\n--- Stage Summary ---")
    print(f"Equivalent Theoretical Stages (Neq) = {Neq}")
    print(f"Real Stages (Nreal) = {Nreal}")
else:
    print("\ n  Cannot compute Neq and Nreal because NT or NB is missing.")

V_B = s * B

# Vapor flow at the top (rectifying section)
V_T = (r + 1) * D

print("\n--- Vapor Rates ---")
print(f"Vapor Rate at Bottom (V_B) = {V_B:.4f} mol/s")
print(f"Vapor Rate at Top (V_T) = {V_T:.4f} mol/s")

lambda_D = 30.8  # Benzene
lambda_toluene = 33.2  # Toluene
lambda_Diphenyl = 56.7  # <-- you need to define this!!

# After calculating x_B_new (bottoms composition), calculate bottoms lambda_B
lambda_B = x_B_new['A'] * lambda_toluene + x_B_new['C'] * lambda_Diphenyl
# Condenser duty (kJ/s)
Q_C = lambda_D * V_T

# Reboiler duty (kJ/s)
Q_R = lambda_toluene * V_B

print("\n--- Heat Duties ---")
print(f"Condenser Duty (Q_C) = {Q_C:.2f} kJ/s")
print(f"Reboiler Duty (Q_R) = {Q_R:.2f} kJ/s")

phi_flood = 0.6
An_over_A = 0.8
c0 = 439  # m/h for tray spacing of 0.61 m
c0_m_per_s = c0 / 3600  # convert to m/s

# Vapor flow rates (from previous calculations)
# Use V_T for rectifying section (top)
# Use V_B for stripping section (bottom)

# Properties
molecular_weights = {
    'Benzene': 78.11,
    'Toluene': 92.14,
    'Diphenyl': 154.21
}

# --- Step 2: Correct densities (g/m ) ---
densities = {
    'Benzene': {'liquid': 876000, 'vapor': 3200},
    'Toluene': {'liquid': 867000, 'vapor': 3100},
    'Diphenyl': {'liquid': 1060000, 'vapor': 5000}
}

# --- Step 3: Calculate Bottoms weighted molecular weight and densities ---
def calculate_bottom_properties(x_B):
    """Calculate molecular weight and densities for bottoms mixture."""
    Mv_bottom = (
        x_B['A'] * molecular_weights['Toluene'] +
        x_B['C'] * molecular_weights['Diphenyl']
    )

    rho_l_bottom = (
        x_B['A'] * densities['Toluene']['liquid'] +
        x_B['C'] * densities['Diphenyl']['liquid']
    )

    rho_v_bottom = (
        x_B['A'] * densities['Toluene']['vapor'] +
        x_B['C'] * densities['Diphenyl']['vapor']
    )
```

```
    return Mv_bottom, rho_l_bottom, rho_v_bottom

# --- Step 4: Area calculation function ---
def calculate_area(Mv, rho_l, rho_v, V):
    """Calculate cross-sectional area A (m  )."""
    term1 = Mv / (math.sqrt(rho_l * rho_v) * phi_flood * c0_m_per_s)
    A = term1 * (1 / An_over_A) * V
    return A  # in m

# Rectifying Section (Benzene vapor at top)
Mv_rect = molecular_weights['Benzene']
rho_l_rect = densities['Benzene']['liquid']
rho_v_rect = densities['Benzene']['vapor']
A_rect = calculate_area(Mv_rect, rho_l_rect, rho_v_rect, V_T)

# Stripping Section (Toluene vapor at bottom)
Mv_strip = molecular_weights['Toluene']
rho_l_strip = densities['Toluene']['liquid']
rho_v_strip = densities['Toluene']['vapor']
A_strip = calculate_area(Mv_strip, rho_l_strip, rho_v_strip, V_B)

print("\n--- Cross-Section Area for Benzene-Toluene Separation ---")
print(f"Rectifying Section Area (Benzene vapor): {A_rect:.4f} m  ")
print(f"Stripping Section Area (Toluene vapor): {A_strip:.4f} m  ")

Ht_inch = 24  # tray spacing in inches
Ht_m = Ht_inch * 0.0254  # convert to meters
Hmin = 3 * Ht_m  # minimum height = 3 * tray spacing

# Number of real stages (from previous calculation)
# Example: Nreal = 35 (you need to already have Nreal computed earlier)
# Nreal must be an integer (already rounded up)

# Calculate total column height
H = Hmin + Ht_m * Nreal

print("\n--- Column Height Estimation ---")
print(f"Tray Spacing (Ht) = {Ht_m:.4f} meters")
print(f"Minimum Height (Hmin) = {Hmin:.4f} meters")
print(f"Total Column Height (H) = {H:.2f} meters")

U = 770  # W/m K
Tavg_range = np.linspace(50, 60, 100)  #  Tavg  from 50 C  to 60 C

# Your previously calculated condenser and reboiler duties in kJ/s
# Make sure you already have Q_C and Q_R from previous steps
# (Example placeholders     replace with your real numbers if needed)
# Q_C = ...
# Q_R = ...

# Convert Q from kJ/s to W
Q_C_W = Q_C * 1000
Q_R_W = Q_R * 1000

# Calculate heat exchanger area Ah for each  Tavg
A_C = Q_C_W / (U * Tavg_range)  # Condenser area
A_R = Q_R_W / (U * Tavg_range)  # Reboiler area

# Key points to highlight
Tavg_key_points = [50, 55, 60]
A_C_keys = Q_C_W / (U * np.array(Tavg_key_points))
A_R_keys = Q_R_W / (U * np.array(Tavg_key_points))

# --- Plotting ---
plt.figure(figsize=(9, 6))

# Plot condenser and reboiler areas
plt.plot(Tavg_range, A_C, label='Condenser Area ($A_C$)', linestyle='-', linewidth
    =2)
plt.plot(Tavg_range, A_R, label='Reboiler Area ($A_R$)', linestyle='--', linewidth
    =2)

# Highlight points at Tavg = 50 C , 55 C , 60 C
```

```
plt.scatter(Tavg_key_points, A_C_keys, color='blue', marker='o')
plt.scatter(Tavg_key_points, A_R_keys, color='orange', marker='x')

# Label the points
for i, T in enumerate(Tavg_key_points):
    plt.text(T, A_C_keys[i] * 1.02, f'{A_C_keys[i]:.2f} m  ', color='blue', ha='
        center')
    plt.text(T, A_R_keys[i] * 1.02, f'{A_R_keys[i]:.2f} m  ', color='orange', ha='
        center')

# Final plot settings
plt.xlabel('Average Temperature Difference ($\Delta T_{avg}$) [ C ]', fontsize=12)
plt.ylabel('Heat Exchanger Area ($A_h$) [m  ]', fontsize=12)
plt.title('Condenser and Reboiler Area vs. Average Temperature Difference', fontsize
    =14)
plt.grid(True)
plt.legend(fontsize=12)
plt.tight_layout()
plt.show()

# Given constants
# Diameter calculations
D_rect = math.sqrt(4 * A_rect / math.pi)  # Rectifying section diameter
D_strip = math.sqrt(4 * A_strip / math.pi)  # Stripping section diameter

print("\n--- Column Diameters ---")
print(f"Rectifying Section Diameter (D_rect) = {D_rect:.4f} meters")
print(f"Stripping Section Diameter (D_strip) = {D_strip:.4f} meters")

# --- Step: Split Total Height into Rectifying and Stripping Heights ---

H_rect = H * NT / Neq
H_strip = H * NB / Neq

print("\n--- Column Height Split ---")
print(f"Rectifying Section Height (H_rect) = {H_rect:.2f} meters")
print(f"Stripping Section Height (H_strip) = {H_strip:.2f} meters")

# --- Step: Calculate Total Installed Costs for Distillation Column (Rectifying +
    Stripping Sections Combined) ---

# Constants
CEPCI = 800
Fs = 1.0  # Tray spacing factor (24 inch tray spacing)

# Unit conversion
meter_to_feet = 3.28084
D_rect_ft = D_rect * meter_to_feet
D_strip_ft = D_strip * meter_to_feet
H_rect_ft = H_rect * meter_to_feet
H_strip_ft = H_strip * meter_to_feet

# Tray type factors (Ft)
tray_types = {
    'Grid': 0.0,
    'Plate': 0.0,
    'Sieve': 0.0,
    'Trough or Valve': 0.4,
    'Bubble Cap': 1.8,
    'Koch Kascade': 3.9
}

# Tray material factors (Fm)
tray_materials = {
    'CS': 0.0,
    'SS': 1.7,
    'Monel': 8.9
}

# Function to calculate Installed Cost (using FEET)
def installed_cost(D_ft, H_ft, Fc):
    base = (CEPCI / 113.7) * 4.7 * D_ft**1.55 * H_ft
    return base * Fc
```

```
# Store results
total_costs = []
labels = []

# Calculate for all combinations
for tray_type, Ft in tray_types.items():
    for tray_material, Fm in tray_materials.items():
        Fc = Fs + Ft + Fm

        # Installed Cost for Rectifying Section
        cost_rect = installed_cost(D_rect_ft, H_rect_ft, Fc)

        # Installed Cost for Stripping Section
        cost_strip = installed_cost(D_strip_ft, H_strip_ft, Fc)

        # Total Installed Cost (Rectifying + Stripping)
        total_cost = cost_rect + cost_strip

        total_costs.append(total_cost)
        labels.append(f"{tray_type} + {tray_material}")

# --- Plotting Total Installed Costs ---
plt.figure(figsize=(10, 8))
bars = plt.bar(labels, total_costs, color='lightseagreen', edgecolor='black')

plt.xticks(rotation=45, ha='right', fontsize=10)
plt.ylabel('Total Installed Cost (USD)', fontsize=12)

# Add exact cost labels on top of each bar
for bar, cost in zip(bars, total_costs):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2.0, height * 1.01, f"${cost:,.0f}", ha
        ='center', va='bottom', fontsize=8)

plt.tight_layout()
plt.show()

CEPCI = 800
m2_to_ft2 = 10.7639

# Assume you already have A_C and A_R arrays from previous condenser and reboiler
    area calculations (in m )

# Convert condenser and reboiler areas to ft
A_C_ft2 = A_C * m2_to_ft2
A_R_ft2 = A_R * m2_to_ft2

# Factors for Design Types and Materials
Fd_condenser = 1.00    # Floating Head
Fd_reboiler = 1.35     # Kettle Reboiler
Fp = 0.0               # Design Pressure up to 150 psi

# Shell-and-tube Material Factors
materials = {
    'CS/CS': 1.00,
    'SS/SS': 3.75,
    'CS/Monel': 3.10,
    'Monel/Monel': 4.25
}

# Cost calculation functions
def purchased_cost(A, Fc):
    return (CEPCI / 113.7) * (101.3 * A**0.65 * Fc)

def installed_cost(A, Fc):
    return (CEPCI / 113.7) * (101.3 * A**0.65) * (2.29 + Fc)

# Store results
cost_curves = {}

plt.figure(figsize=(10, 8))
```

```
for material, Fm in materials.items():
    Fc_C = (Fd_condenser + Fp) * Fm
    Fc_R = (Fd_reboiler + Fp) * Fm

    cost_C = installed_cost(A_C_ft2, Fc_C)
    cost_R = installed_cost(A_R_ft2, Fc_R)
    total_cost = cost_C + cost_R

    plt.plot(Tavg_range, total_cost, label=material, linewidth=2)

    # Label lowest cost
    idx_min = np.argmin(total_cost)
    T_min = Tavg_range[idx_min]
    cost_min = total_cost[idx_min]
    plt.text(T_min, cost_min, f"${cost_min:,.0f}", fontsize=9, ha='left', va='bottom
        ')

plt.xlabel('Average Temperature Difference ($\Delta T_{avg}$) [ C ]', fontsize=12)
plt.ylabel('Total Installed Cost (USD)', fontsize=12)
plt.legend(title='Material Combination')
plt.tight_layout()
plt.show()
```

8.7.3 Binary Distillation Using Underwood Equation

```
import numpy as np
import math
import matplotlib.pyplot as plt


# --- Step 2: Feed and composition ---
F = 4.039  # mol/s
z = {'A': 0.6055, 'C': 0.3945}  # A: Toluene, C: Diphenyl

alpha = {
    'A':27.1543,
    'C': 1.0
}

alpha_vals = sorted(alpha.values())

# --- Step 3: Flow rates and specifications ---
D = F * z['A']
B = F - D

print(f"Feed Flowrate (F) = {F:.4f} mol/s")
print(f"Distillate Flowrate (D) = {D:.4f} mol/s")
print(f"Bottoms Flowrate (B) = {B:.4f} mol/s")

x_D = {'A': 0.999, 'C': 0.001}
x_B = {
    'A': (F * z['A'] - D * x_D['A']) / B,
    'C': (F * z['C'] - D * x_D['C']) / B
}

print("--- Distillate Composition (x_D) ---")
for k, v in x_D.items():
    print(f"{k}: {v:.4f}")

print("--- Bottom Composition (x_B) ---")
for k, v in x_B.items():
    print(f"{k}: {v:.4f}")

# --- Step 4: Underwood equation ---
def underwood_eq(theta):
    return sum((alpha[i] * z[i]) / (alpha[i] - theta) for i in alpha)

def bisection(f, a, b, tol=1e-10, max_iter=1000):
    fa, fb = f(a), f(b)
    if fa * fb > 0:
        return None
    for _ in range(max_iter):
```

```
        c = (a + b) / 2
        fc = f(c)
        if abs(fc) < tol or (b - a) / 2 < tol:
            return c
        if fa * fc < 0:
            b, fb = c, fc
        else:
            a, fa = c, fc
    return None

# Find Underwood root
theta1 = None
for i in range(len(alpha_vals) - 1):
    a = alpha_vals[i] + 1e-4
    b = alpha_vals[i + 1] - 1e-4
    root = bisection(underwood_eq, a, b)
    if root is not None:
        theta1 = root
        break

print(f"Underwood Root (theta1) = {theta1:.6f}")

# --- Step 5: Calculate minimum reflux ratio ---
rmin = sum((alpha[i] * x_D[i]) / (alpha[i] - theta1) for i in alpha) - 1
print(f"Minimum Reflux Ratio (rmin) = {rmin:.6f}")

# --- Step 6: Operational Boil-Up Ratio ---
r = 2 * rmin
q = 1.0
D_to_B = D / B
s = D_to_B * (r + q) - (1 - q)
smin = D_to_B*(rmin +q)-(1-q)
print(f"Operational Boil-Up Ratio (s) = {s:.6f}")

# --- Step 7: Theoretical stages ---
# --- Step 7: Solve for feed stage composition x_f ---
x_f = {}
for k in alpha:
    x_f[k] = (q * z[k] + (1 - q) * x_D[k])

print("--- Feed Stage Composition (x_f) ---")
for k, v in x_f.items():
    print(f"{k}: {v:.4f}")
# --- Step 8: Theoretical stages ---
# --- Step 8: Rectifying Section Roots (solve using x_D) ---

# --- Find rectifying roots ---
def rectifying_underwood_eq(phi):
    try:
        return sum((alpha[i] * x_D[i]) / (alpha[i] - phi) for i in alpha) - (r + 1)
    except ZeroDivisionError:
        return float('inf')

def stripping_underwood_eq(phi):
    try:
        return s + sum((alpha[i] * x_B[i]) / (alpha[i] - phi) for i in alpha)
    except ZeroDivisionError:
        return float('inf')

# --- Bisection method ---
def bisection1(f, a, b, tol=1e-10, max_iter=1000):
    fa, fb = f(a), f(b)
    if fa * fb > 0:
        return None
    for _ in range(max_iter):
        c = (a + b) / 2
        fc = f(c)
        if abs(fc) < tol or (b - a) / 2 < tol:
            return c
        if fa * fc < 0:
            b, fb = c, fc
        else:
            a, fa = c, fc
```

```
    return None

# --- Root finder ---
def find_multiple_roots_safe(f, alpha_vals, tol=1e-10):
    roots = []
    offset = 1e-3
    sorted_alpha = sorted(alpha_vals)

    intervals = []
    intervals.append((min(sorted_alpha) - 2, min(sorted_alpha) - offset))
    for i in range(len(sorted_alpha) - 1):
        intervals.append((sorted_alpha[i] + offset, sorted_alpha[i+1] - offset))
    intervals.append((max(sorted_alpha) + offset, max(sorted_alpha) + 2))

    for (a, b) in intervals:
        test_points = np.linspace(a, b, 1000)
        for i in range(len(test_points) - 1):
            x0, x1 = test_points[i], test_points[i + 1]
            try:
                if f(x0) * f(x1) < 0:
                    root = bisection1(f, x0, x1, tol)
                    if root is not None and all(abs(root - existing) > 1e-6 for
                        existing in roots):
                        roots.append(root)
            except:
                continue
    return sorted(roots, reverse=True)

# --- NT and NB Calculation ---
def calculate_NT(alpha, x_f, phi1, phi2):
    numerator = (alpha['A'] / (alpha['A'] - phi2)) * x_f['A'] + (alpha['C'] / (alpha
        ['C'] - phi2)) * x_f['C']
    denominator = (alpha['A'] / (alpha['A'] - phi1)) * x_f['A'] + (alpha['C'] / (
        alpha['C'] - phi1)) * x_f['C']
    if numerator/denominator> 0 and phi1 > phi2:
        return math.log(numerator / denominator) / math.log(phi1 / phi2)
    else:
        return None

def calculate_NB(alpha, x_f, phi1, phi2):
    numerator = (alpha['A'] / (alpha['A'] - phi1)) * x_f['A'] + (alpha['C'] / (alpha
        ['C'] - phi1)) * x_f['C']
    denominator = (alpha['A'] / (alpha['A'] - phi2)) * x_f['A'] + (alpha['C'] / (
        alpha['C'] - phi2)) * x_f['C']
    print("NB numerator:", numerator)
    print("NB denominator:", denominator)
    print("log(phi1/phi2):", math.log(phi1 / phi2))
    if numerator /denominator >0 and phi1 > phi2:
        return math.log(numerator / denominator) / math.log(phi1 / phi2)
    else:
        return None

# --- Execute Workflow ---
rectifying_roots = find_multiple_roots_safe(rectifying_underwood_eq, alpha_vals)
stripping_roots = find_multiple_roots_safe(stripping_underwood_eq, alpha_vals)

print("\n--- Rectifying Section Roots ---")
for i, root in enumerate(rectifying_roots):
    print(f"Rectifying Root {i+1}:    = {root:.6f}")

print("\n--- Stripping Section Roots ---")
for i, root in enumerate(stripping_roots):
    print(f"Stripping Root {i+1}:    = {root:.6f}")

# --- Calculate NT from stored rectifying roots ---
print("\n--- Rectifying Section NT Calculations ---")
if len(rectifying_roots) >= 2:
    phi1 = rectifying_roots[0]
    phi2 = rectifying_roots[1]
    NT = calculate_NT(alpha, x_f, phi1, phi2)
    print(f"      = {phi1:.6f},      = {phi2:.6f}    NT = {NT}")
else:
    print("Not enough rectifying roots to calculate NT.")
```

```
# --- Calculate NB from stored stripping roots ---
print("\n--- Stripping Section NB Calculations ---")
if len(stripping_roots) >= 2:
    phi1 = stripping_roots[0]
    phi2 = stripping_roots[1]
    NB = calculate_NB(alpha, x_f, phi1, phi2)
    print(f"     =  {phi1:.6f},      =  {phi2:.6f}    NB =  {NB}")
else:
    print("Not enough stripping roots to calculate NB.")

if 'NT' in locals() and 'NB' in locals() and NT is not None and NB is not None:
    Neq = NT + NB+1
    Nreal = math.ceil(Neq / 0.3)  # Always round up
    print("\n--- Stage Summary ---")
    print(f"Equivalent Theoretical Stages (Neq) = {Neq}")
    print(f"Real Stages (Nreal) = {Nreal}")
else:
    print("\ n  Cannot compute Neq and Nreal because NT or NB is missing.")

V_B = s * B

# Vapor flow at the top (rectifying section)
V_T = (r + 1) * D

print("\n--- Vapor Rates ---")
print(f"Vapor Rate at Bottom (V_B) = {V_B:.4f} mol/s")
print(f"Vapor Rate at Top (V_T) = {V_T:.4f} mol/s")

# --- Step 11 (for Toluene and Biphenyl): Calculate Condenser and Reboiler Duties QC
    and QR ---

# Latent heats in kJ/mol
lambda_D = 33.2  # Toluene
lambda_B = 56.7  # Biphenyl

# Condenser duty (kJ/s)
Q_C = lambda_D * V_T

# Reboiler duty (kJ/s)
Q_R = lambda_B * V_B

print("\n--- Heat Duties (Toluene at Top, Biphenyl at Bottom) ---")
print(f"Condenser Duty (Q_C) = {Q_C:.2f} kJ/s")
print(f"Reboiler Duty (Q_R) = {Q_R:.2f} kJ/s")

phi_flood = 0.6
An_over_A = 0.8
c0 = 439  # m/h for tray spacing of 0.61 m
c0_m_per_s = c0 / 3600  # convert to m/s

# Vapor flow rates (from previous calculations)
# Use V_T for rectifying section (top)
# Use V_B for stripping section (bottom)

# Properties
molecular_weights = {
    'Benzene': 78.11,
    'Toluene': 92.14,
    'Diphenyl': 154.21
}

# --- Step 2: Correct densities (g/m  ) ---
densities = {
    'Benzene': {'liquid': 876000, 'vapor': 3200},
    'Toluene': {'liquid': 867000, 'vapor': 3100},
    'Diphenyl': {'liquid': 1060000, 'vapor': 5000}
}

# --- Step 3: Calculate Bottoms weighted molecular weight and densities ---
def calculate_bottom_properties(x_B):
    """Calculate molecular weight and densities for bottoms mixture."""
    Mv_bottom = (
```

```
        x_B['A'] * molecular_weights['Toluene'] +
        x_B['C'] * molecular_weights['Diphenyl']
    )

    rho_l_bottom = (
        x_B['A'] * densities['Toluene']['liquid'] +
        x_B['C'] * densities['Diphenyl']['liquid']
    )

    rho_v_bottom = (
        x_B['A'] * densities['Toluene']['vapor'] +
        x_B['C'] * densities['Diphenyl']['vapor']
    )

    return Mv_bottom, rho_l_bottom, rho_v_bottom

# --- Step 4: Area calculation function ---
def calculate_area(Mv, rho_l, rho_v, V):
    """Calculate cross-sectional area A (m )."""
    term1 = Mv / (math.sqrt(rho_l * rho_v) * phi_flood * c0_m_per_s)
    A = term1 * (1 / An_over_A) * V
    return A  # in m
# Rectifying Section (Toluene vapor at top)
Mv_rect = molecular_weights['Benzene']
rho_l_rect = densities['Benzene']['liquid']
rho_v_rect = densities['Benzene']['vapor']
A_rect = calculate_area(Mv_rect, rho_l_rect, rho_v_rect, V_T)

# --- Stripping Section (bottoms mixture: Toluene + Diphenyl) ---
Mv_strip, rho_l_strip, rho_v_strip = calculate_bottom_properties(x_B_new)
A_strip = calculate_area(Mv_strip, rho_l_strip, rho_v_strip, V_B)

print("\n--- Cross-Section Area for Toluene-Biphenyl Separation ---")
print(f"Rectifying Section Area (Toluene vapor): {A_rect:.4f} m  ")
print(f"Stripping Section Area (Biphenyl vapor): {A_strip:.4f} m  ")

Ht_inch = 24  # tray spacing in inches
Ht_m = Ht_inch * 0.0254  # convert to meters
Hmin = 3 * Ht_m  # minimum height = 3 * tray spacing

# Number of real stages (from previous calculation)
# Example: Nreal = 35 (you need to already have Nreal computed earlier)
# Nreal must be an integer (already rounded up)

# Calculate total column height
H = Hmin + Ht_m * Nreal

print("\n--- Column Height Estimation ---")
print(f"Tray Spacing (Ht) = {Ht_m:.4f} meters")
print(f"Minimum Height (Hmin) = {Hmin:.4f} meters")
print(f"Total Column Height (H) = {H:.2f} meters")
U = 770  # W/m K
Tavg_range = np.linspace(50, 60, 100)  #  Tavg  from 50 C  to 60 C

# Your previously calculated condenser and reboiler duties in kJ/s
# Make sure you already have Q_C and Q_R from previous steps
# (Example placeholders     replace with your real numbers if needed)
# Q_C = ...
# Q_R = ...

# Convert Q from kJ/s to W
Q_C_W = Q_C * 1000
Q_R_W = Q_R * 1000

# Calculate heat exchanger area Ah for each  Tavg
A_C = Q_C_W / (U * Tavg_range)  # Condenser area
A_R = Q_R_W / (U * Tavg_range)  # Reboiler area

# Key points to highlight
Tavg_key_points = [50, 55, 60]
A_C_keys = Q_C_W / (U * np.array(Tavg_key_points))
A_R_keys = Q_R_W / (U * np.array(Tavg_key_points))
```

```
# --- Plotting ---
plt.figure(figsize=(9, 6))

# Plot condenser and reboiler areas
plt.plot(Tavg_range, A_C, label='Condenser Area ($A_C$)', linestyle='-', linewidth
    =2)
plt.plot(Tavg_range, A_R, label='Reboiler Area ($A_R$)', linestyle='--', linewidth
    =2)

# Highlight points at Tavg = 50 C , 55 C , 60 C
plt.scatter(Tavg_key_points, A_C_keys, color='blue', marker='o')
plt.scatter(Tavg_key_points, A_R_keys, color='orange', marker='x')

# Label the points
for i, T in enumerate(Tavg_key_points):
    plt.text(T, A_C_keys[i] * 1.02, f'{A_C_keys[i]:.2f} m  ', color='blue', ha='
        center')
    plt.text(T, A_R_keys[i] * 1.02, f'{A_R_keys[i]:.2f} m  ', color='orange', ha='
        center')

# Final plot settings
plt.xlabel('Average Temperature Difference ($\Delta T_{avg}$) [ C ]', fontsize=12)
plt.ylabel('Heat Exchanger Area ($A_h$) [m  ]', fontsize=12)
plt.title('Condenser and Reboiler Area vs. Average Temperature Difference', fontsize
    =14)
plt.grid(True)
plt.legend(fontsize=12)
plt.tight_layout()
plt.show()

# Diameter calculations
D_rect = math.sqrt(4 * A_rect / math.pi)  # Rectifying section diameter
D_strip = math.sqrt(4 * A_strip / math.pi)  # Stripping section diameter

print("\n--- Column Diameters ---")
print(f"Rectifying Section Diameter (D_rect) = {D_rect:.4f} meters")
print(f"Stripping Section Diameter (D_strip) = {D_strip:.4f} meters")

# --- Step: Split Total Height into Rectifying and Stripping Heights ---

# Make sure Neq = NT + NB (already computed earlier)
H_rect = H * NT / Neq
H_strip = H * NB / Neq

print("\n--- Column Height Split ---")
print(f"Rectifying Section Height (H_rect) = {H_rect:.2f} meters")
print(f"Stripping Section Height (H_strip) = {H_strip:.2f} meters")

# --- Step: Calculate Total Installed Costs for Distillation Column (Rectifying +
    Stripping Sections Combined) ---

# Constants
CEPCI = 800
Fs = 1.0  # Tray spacing factor (24 inch tray spacing)

# Unit conversion
meter_to_feet = 3.28084
D_rect_ft = D_rect * meter_to_feet
D_strip_ft = D_strip * meter_to_feet
H_rect_ft = H_rect * meter_to_feet
H_strip_ft = H_strip * meter_to_feet

# Tray type factors (Ft)
tray_types = {
    'Grid': 0.0,
    'Plate': 0.0,
    'Sieve': 0.0,
    'Trough or Valve': 0.4,
    'Bubble Cap': 1.8,
    'Koch Kascade': 3.9
}

# Tray material factors (Fm)
```

```
tray_materials = {
    'CS': 0.0,
    'SS': 1.7,
    'Monel': 8.9
}

# Function to calculate Installed Cost (using FEET)
def installed_cost(D_ft, H_ft, Fc):
    base = (CEPCI / 113.7) * 4.7 * D_ft**1.55 * H_ft
    return base * Fc

# Store results
total_costs = []
labels = []

# Calculate for all combinations
for tray_type, Ft in tray_types.items():
    for tray_material, Fm in tray_materials.items():
        Fc = Fs + Ft + Fm

        # Installed Cost for Rectifying Section
        cost_rect = installed_cost(D_rect_ft, H_rect_ft, Fc)

        # Installed Cost for Stripping Section
        cost_strip = installed_cost(D_strip_ft, H_strip_ft, Fc)

        # Total Installed Cost (Rectifying + Stripping)
        total_cost = cost_rect + cost_strip

        total_costs.append(total_cost)
        labels.append(f"{tray_type} + {tray_material}")

# --- Plotting Total Installed Costs ---
plt.figure(figsize=(14, 8))
bars = plt.bar(labels, total_costs, color='lightseagreen', edgecolor='black')

plt.xticks(rotation=45, ha='right', fontsize=10)
plt.ylabel('Total Installed Cost (USD)', fontsize=12)
plt.title('Total Installed Cost for Different Tray Type and Material Combinations',
    fontsize=14)

# Add exact cost labels on top of each bar
for bar, cost in zip(bars, total_costs):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2.0, height * 1.01, f"${cost:,.0f}", ha
        ='center', va='bottom', fontsize=8)

plt.tight_layout()
plt.show()

CEPCI = 800
m2_to_ft2 = 10.7639

# Assume you already have A_C and A_R arrays from previous condenser and reboiler
    area calculations (in m )

# Convert condenser and reboiler areas to ft
A_C_ft2 = A_C * m2_to_ft2
A_R_ft2 = A_R * m2_to_ft2

# Factors for Design Types and Materials
Fd_condenser = 1.00    # Floating Head
Fd_reboiler = 1.35     # Kettle Reboiler
Fp = 0.0               # Design Pressure up to 150 psi

# Shell-and-tube Material Factors
materials = {
    'CS/CS': 1.00,
    'SS/SS': 3.75,
    'CS/Monel': 3.10,
    'Monel/Monel': 4.25
}
```

```
# Cost calculation functions
def purchased_cost(A, Fc):
    return (CEPCI / 113.7) * (101.3 * A**0.65 * Fc)

def installed_cost(A, Fc):
    return (CEPCI / 113.7) * (101.3 * A**0.65) * (2.29 + Fc)

# Store results
cost_curves = {}

plt.figure(figsize=(10, 8))

for material, Fm in materials.items():
    Fc_C = (Fd_condenser + Fp) * Fm
    Fc_R = (Fd_reboiler + Fp) * Fm

    cost_C = installed_cost(A_C_ft2, Fc_C)
    cost_R = installed_cost(A_R_ft2, Fc_R)
    total_cost = cost_C + cost_R

    plt.plot(Tavg_range, total_cost, label=material, linewidth=2)

    # Label lowest cost
    idx_min = np.argmin(total_cost)
    T_min = Tavg_range[idx_min]
    cost_min = total_cost[idx_min]
    plt.text(T_min, cost_min, f"${cost_min:,.0f}", fontsize=9, ha='left', va='bottom
        ')

plt.xlabel('Average␣Temperature␣Difference␣($\Delta␣T_{avg}$)␣[ C ]', fontsize=12)
plt.ylabel('Total␣Installed␣Cost␣(USD)', fontsize=12)
plt.title('Total␣Installed␣Cost␣vs.␣Temperature␣Difference␣for␣Different␣Materials',
     fontsize=14)
plt.legend(title='Material␣Combination')
plt.tight_layout()
plt.show()
```

8.7.4 Heat Exchanger Optimization

```
import numpy as np
import matplotlib.pyplot as plt

# === CONFIGURATION ===
KW_TO_MW = 1.0 / 1000.0
DT_MIN = 10.0  #  C
N_POINTS = 1001
T_HOT_RANGE = (25, 580)
T_COLD_RANGE = (669, 50)

# === INPUT DATA ===
mass_flows = {
    'H2': 1969.0,
    'Toluene': 18180.0,
}

product_composition = {
    'H2': 0.0578,
    'CH4': 0.1953,
    'Toluene': 0.0578,
    'Benzene': 0.5996,
    'Diphenyl': 0.0557,
}

boiling_points_celsius = {
    'H2': -252.87,
    'Toluene': 110.6,
    'CH4': -161.5,
    'Benzene': 80.1,
    'Diphenyl': 255.0,
}

latent_heats = {
```

```
    'H2': 447.0,
    'Toluene': 351.0,
    'CH4': 510.0,
    'Benzene': 394.0,
    'Diphenyl': 332.0,
}

MW = {
    'H2': 2.016e-3,
    'CH4': 16.04e-3,
    'Toluene': 92.14e-3,
    'Benzene': 78.11e-3,
    'Diphenyl': 154.22e-3,
}

def convert_cp(Amol, Bmol, Cmol, Dmol, MW):
    return [Amol / 1e3 / MW, Bmol / 1e3 / MW, Cmol / 1e3 / MW, Dmol / 1e3 / MW]

Cp_coeffs = {
    'H2': convert_cp(29.553, 7.64e-3, -8.934e-6, 2.447e-9, MW['H2']),
    'CH4': convert_cp(19.996, 5.024e-2, 1.269e-5, -1.101e-8, MW['CH4']),
    'Toluene': convert_cp(111.331, 1.088e-1, -2.266e-4, 3.729e-7, MW['Toluene']),
    'Benzene': convert_cp(82.141, 1.3408e-1, 3.708e-5, 5.459e-8, MW['Benzene']),
    'Diphenyl': convert_cp(224.92, -1.2495e-1, 2.895e-4, -7.216e-7, MW['Diphenyl']),
}

T_boil_Toluene = boiling_points_celsius['Toluene']

def Cp_pure(T, ABCD):
    A, B, C, D = ABCD
    T = np.asarray(T)
    return A + B*T + C*T**2 + D*T**3

def calculate_duty_with_latent_heat(T_values, streams, boiling_pts, latent_heats,
    Cp_dict):
    n = len(T_values)
    Q = np.zeros(n)
    for i in range(n - 1):
        T1, T2 = T_values[i], T_values[i+1]
        dT = T2 - T1
        T_mid = 0.5 * (T1 + T2)

        for m_hr, key, y, *temp_range in streams:
            if key not in Cp_dict: continue
            m_s = (m_hr / 3600.0) * (y if y is not None else 1.0)
            T_start, T_end = temp_range if temp_range else (-np.inf, np.inf)
            if T_start <= T_mid <= T_end:
                Q[i+1] += Cp_pure(T_mid, Cp_dict[key]) * m_s * dT

        for m_hr, key, y, *temp_range in streams:
            if key not in latent_heats or key not in boiling_pts: continue
            m_s = (m_hr / 3600.0) * (y if y is not None else 1.0)
            T_boil = boiling_pts[key]
            T_start, T_end = temp_range if temp_range else (-np.inf, np.inf)
            if T_start <= T_boil <= T_end:
                if T1 < T_boil <= T2:
                    Q[i+1] += latent_heats[key] * m_s
                elif T2 < T_boil <= T1:
                    Q[i+1] -= latent_heats[key] * m_s

        Q[i+1] += Q[i]
    return Q

def get_composite_curves():
    T_hot = np.linspace(*T_HOT_RANGE, N_POINTS)
    T_cold = np.linspace(*T_COLD_RANGE, N_POINTS)

    hot_streams = [
        (mass_flows['H2'], 'H2', None, 25, 580),
        (mass_flows['Toluene'], 'Toluene', None, 25, 580),
    ]
    cold_streams = [
        (sum(mass_flows.values()), comp, y, 50, 669)
```

```
        for comp, y in product_composition.items()
    ]

    Q_hot_kw = calculate_duty_with_latent_heat(T_hot, hot_streams,
        boiling_points_celsius, latent_heats, Cp_coeffs)
    Q_cold_kw_rev = calculate_duty_with_latent_heat(T_cold, cold_streams,
        boiling_points_celsius, latent_heats, Cp_coeffs)
    Q_hot = Q_hot_kw * KW_TO_MW
    Q_cold = np.flip(Q_cold_kw_rev * KW_TO_MW) - Q_cold_kw_rev[-1] * KW_TO_MW
    return T_hot, T_cold[::-1], Q_hot, Q_cold

def find_pinch_offset(Q_hot, Q_cold, T_hot, T_cold):
    T_target = T_boil_Toluene - DT_MIN
    Q_cold_at_pinch = np.interp(T_target, T_cold, Q_cold)
    Q_hot_at_boil = np.interp(T_boil_Toluene, T_hot, Q_hot)
    return Q_cold_at_pinch - Q_hot_at_boil

def plot_composite_curves(Q_hot, Q_cold, T_hot, T_cold, Q_offset, Q_plot_shift=1.82)
    :
    Q_hot_shifted = Q_hot + Q_offset + Q_plot_shift

    plt.figure(figsize=(10,7))
    plt.plot(Q_hot_shifted, T_hot, label='Hot␣Stream␣(Shifted)', lw=2)
    plt.plot(Q_cold, T_cold, label='Cold␣Stream', lw=2)

    Q_cool_start = Q_cold[0]
    Q_cool_end = Q_offset + Q_plot_shift
    T_arrow_cool = T_cold[0] - 40
    plt.annotate('', xy=(Q_cool_start, T_arrow_cool), xytext=(Q_cool_end,
        T_arrow_cool),
                 arrowprops=dict(arrowstyle='<->', lw=1.5, color='black'))
    plt.text((Q_cool_start + Q_cool_end) / 2, T_arrow_cool - 10, r'$Q_{cool}$',
        fontsize=12, ha='center')

    Q_hot_start = Q_cold[-1]
    Q_hot_end = Q_hot_shifted[-1]
    T_arrow_hot = T_hot[-1] + 40
    plt.annotate('', xy=(Q_hot_start, T_arrow_hot), xytext=(Q_hot_end, T_arrow_hot),
                 arrowprops=dict(arrowstyle='<->', lw=1.5, color='black'))
    plt.text((Q_hot_start + Q_hot_end) / 2, T_arrow_hot + 10, r'$Q_{hot}$', fontsize
        =12, ha='center')

    plt.xlabel("Heat␣Duty␣Q␣(MW)", fontsize=14)
    plt.ylabel("Temperature␣( C )", fontsize=14)
    plt.legend(fontsize=12)
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()

# === MAIN EXECUTION ===
T_hot, T_cold, Q_hot, Q_cold = get_composite_curves()
Q_offset = find_pinch_offset(Q_hot, Q_cold, T_hot, T_cold)
plot_composite_curves(Q_hot, Q_cold, T_hot, T_cold, Q_offset)

# === UTILITY DUTIES ===
Q_plot_shift = 1.82
Q_hot_shifted = Q_hot + Q_offset +Q_plot_shift
Q_cold_utility = Q_offset
Q_exchange = Q_cold[-1] - Q_offset
Q_hot_utility = Q_hot_shifted[-1] - Q_cold[-1]

Q_cold_utility = max(0, Q_cold_utility)
Q_exchange = max(0, Q_exchange)
Q_hot_utility = max(0, Q_hot_utility)

print("\n===␣Heat␣Integration␣Summary␣===")
print(f"Cooling␣Utility␣Required:␣␣{Q_cold_utility:.2f}␣MW␣(below␣pinch)")
print(f"␣Heat␣Exchanged␣Internally:␣{Q_exchange:.2f}␣MW␣(overlap␣region)")
print(f"␣Heating␣Utility␣Required:␣␣{Q_hot_utility:.2f}␣MW␣(above␣pinch)")

# === NO HEAT EXCHANGER CASE ===
T_feed_range = np.linspace(25, 580, N_POINTS)
T_product_range = np.linspace(669, 50, N_POINTS)
```

```
feed_streams = [
    (mass_flows['H2'], 'H2', 1.0),
    (mass_flows['Toluene'], 'Toluene', 1.0),
]
product_streams = [
    (sum(mass_flows.values()), k, v) for k, v in product_composition.items()
]

def calculate_total_duty(T_values, streams, Cp_dict):
    Q = 0.0
    for i in range(len(T_values) - 1):
        T1, T2 = T_values[i], T_values[i + 1]
        dT = T2 - T1
        T_mid = 0.5 * (T1 + T2)
        for m_hr, key, y in streams:
            if key not in Cp_dict:
                continue
            m_s = (m_hr / 3600.0) * y
            cp = Cp_pure(T_mid, Cp_dict[key])
            Q += cp * m_s * dT
    return Q * KW_TO_MW

Q_heating = calculate_total_duty(T_feed_range, feed_streams, Cp_coeffs)
Q_cooling = calculate_total_duty(T_product_range, product_streams, Cp_coeffs)

print("\n=== Without Heat Exchanger ===")
print(f" Heating Duty: {Q_heating:.2f} MW")
print(f"Cooling Duty: {abs(Q_cooling):.2f} MW")

# === PERCENTAGE SAVINGS ===
heating_saved_pct = 100 * (Q_heating - Q_hot_utility) / Q_heating if Q_heating else
    0
cooling_saved_pct = 100 * (abs(Q_cooling) - Q_cold_utility) / abs(Q_cooling) if
    Q_cooling else 0
print("\n=== Utility Savings with Heat Exchanger ===")
print(f"Heating Duty Saved: {heating_saved_pct:.1f}%")
print(f" Cooling Duty Saved: {cooling_saved_pct:.1f}%")
```