

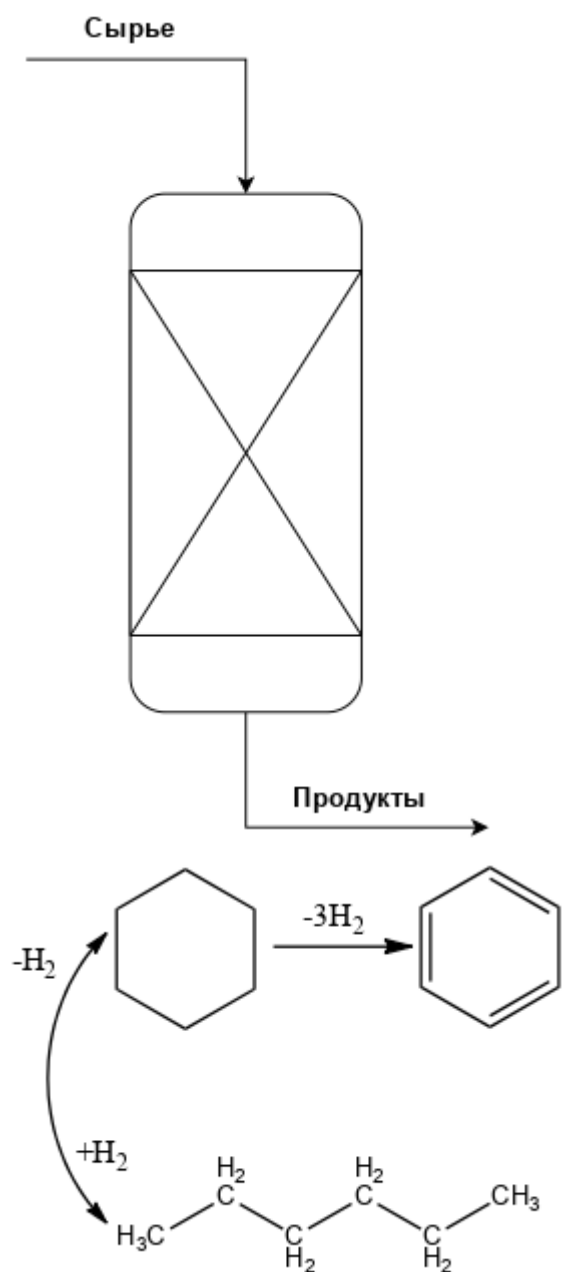
Python для задач химической технологии

Лабораторная работа №7

Введение в объектно-ориентированное программирование

Пример

Рассчитать состав продуктов:



Исходные данные:

Компонент	Концентрация, моль/л
cyclo- C_6H_{12}	0.8
n- C_6H_{14}	0.2
C_6H_6	0.0

Компонент	Концентрация, моль/л
H_2	0.0

Реакции, представленные на схеме:

1. $\text{cyclo-}C_6H_{12} \longrightarrow C_6H_6 + 3H_2$
2. $\text{cyclo-}C_6H_{12} + H_2 \longrightarrow \text{n-}C_6H_{14}$
3. $\text{n-}C_6H_{14} \longrightarrow \text{cyclo-}C_6H_{12} + H_2$

Скорости реакций:

1. $r_1 = k_1 \cdot [\text{cyclo-}C_6H_{12}]$
2. $r_2 = k_2 \cdot [\text{cyclo-}C_6H_{12}] \cdot [H_2]$
3. $r_3 = k_3 \cdot [\text{n-}C_6H_{14}]$

$$\left\{ \begin{array}{l} \frac{d[\text{cyclo-}C_6H_{12}]}{dt} = -r_1 - r_2 + r_3; \\ \frac{d[\text{n-}C_6H_{14}]}{dt} = r_2 - r_3; \\ \frac{d[C_6H_6]}{dt} = r_1; \\ \frac{d[H_2]}{dt} = 3r_1 - r_2 + r_3 \end{array} \right.$$

$$k_1 = 0.4; k_2 = 0.05; k_3 = 0.25$$

Расход сырья 10 л/с. Объем реактора 15 л.

Реализация класса Flow

```
# flow.py
import numpy as np

class Flow:
    def __init__(
        self,
        volume_flow_rate: float,
        molar_fractions: np.ndarray
    ) -> None:
        self.volume_flow_rate = volume_flow_rate
        self.molar_fractions = molar_fractions
        return

if __name__ == '__main__':
    ...
```

Реализация класса Reactor

```
# reactor.py
import numpy as np
```

```

from scipy.integrate import solve_ivp
from typing import Callable
import matplotlib.pyplot as plt
import flow

class Reactor:
    def __init__(self, volume: float) -> None:
        self.volume = volume
        return

    def calculate(self, kinetic_equations: Callable,
                  feedstock: flow.Flow, args: tuple = (), n: int = 50) -> None:

        self.feedstock = feedstock
        self.residence_time = self.volume / self.feedstock.volume_flow_rate
        self.time = np.linspace(0, self.residence_time, n)

        self.solution = solve_ivp(
            fun=kinetic_equations, t_span=(0, self.residence_time),
            y0=self.feedstock.molar_fractions,
            dense_output=True, args=args
        )
        self.products = flow.Flow(
            volume_flow_rate=self.feedstock.volume_flow_rate,
            molar_fractions=self.solution.y[:, -1],
        )
        return

    def draw_profile(self, filename: str = '', labels: list[str] = []) -> None:
        profile = self.solution.sol(self.time)

        if not labels:
            legend = False
            labels = ('', ) * profile.shape[0]
        else:
            legend = True

        for mf, label in zip(profile, labels):
            plt.plot(self.time, mf, label=label)

        if legend:
            plt.legend()

        plt.xlabel('Время, с')
        plt.ylabel('Концентрация компонента, моль/л')
        plt.tight_layout()

        if filename:
            plt.savefig(filename, dpi=800)

        plt.show()
        return

```

```
if __name__ == '__main__':
    ...
```

Реализация кинетических уравнений

```
# kinetic.py
import numpy as np

def equations(t: float, c: np.ndarray, k: np.ndarray) -> np.ndarray:
    cC6H12, nC6H12, C6H6, H2 = c
    k1, k2, k3 = k

    r1 = k1 * cC6H12
    r2 = k2 * cC6H12 * H2
    r3 = k3 * nC6H12

    dcC6H12_dt = -r1 - r2 + r3
    dnC6H12_dt = r2 - r3
    dC6H6_dt = r1
    dH2_dt = 3 * r1 - r2 + r3

    return np.array([dcC6H12_dt, dnC6H12_dt, dC6H6_dt, dH2_dt])

if __name__ == '__main__':
    ...
```

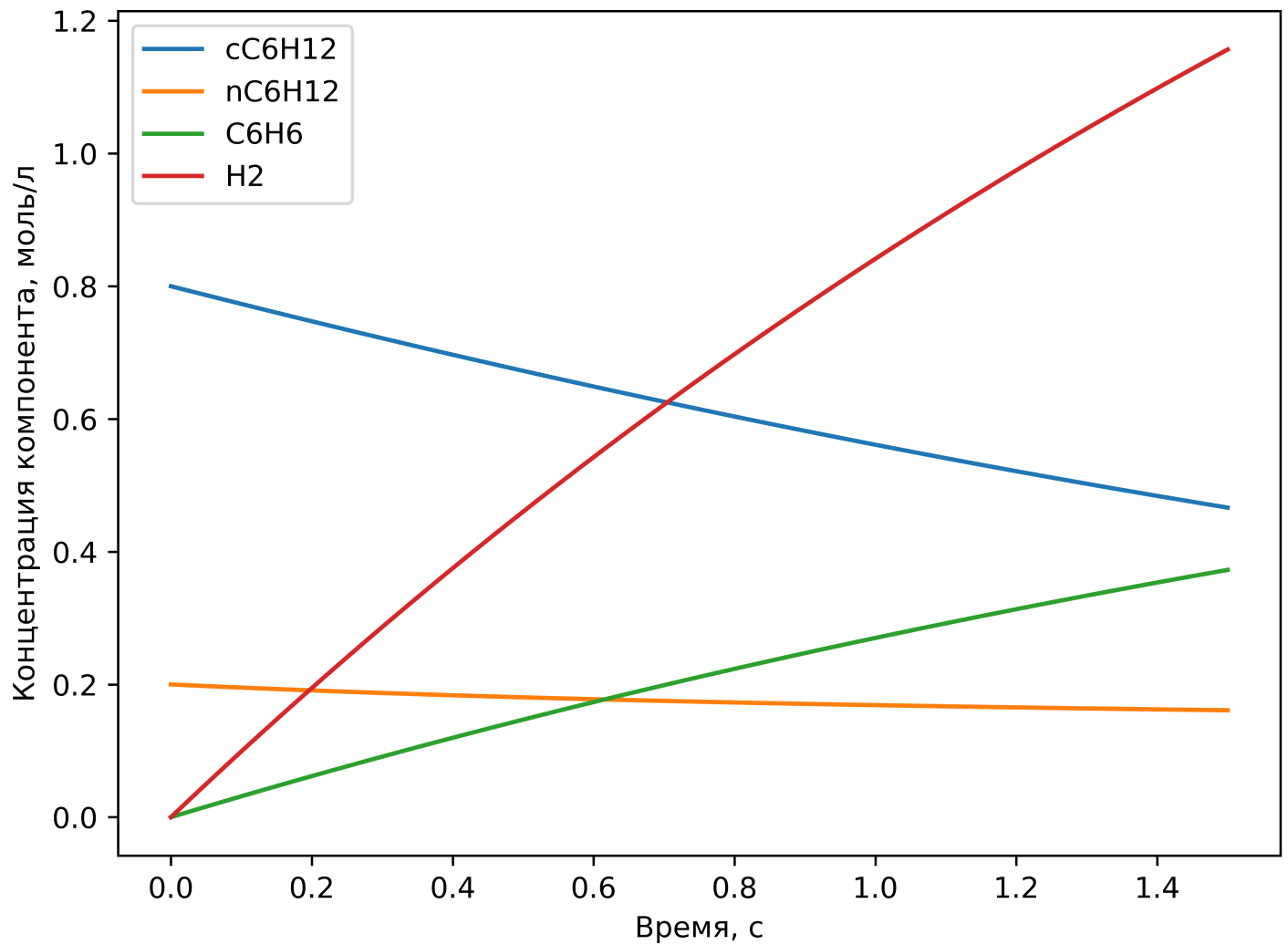
Основная программа:

```
# main.py
import numpy as np
import flow
import reactor
import kinetic

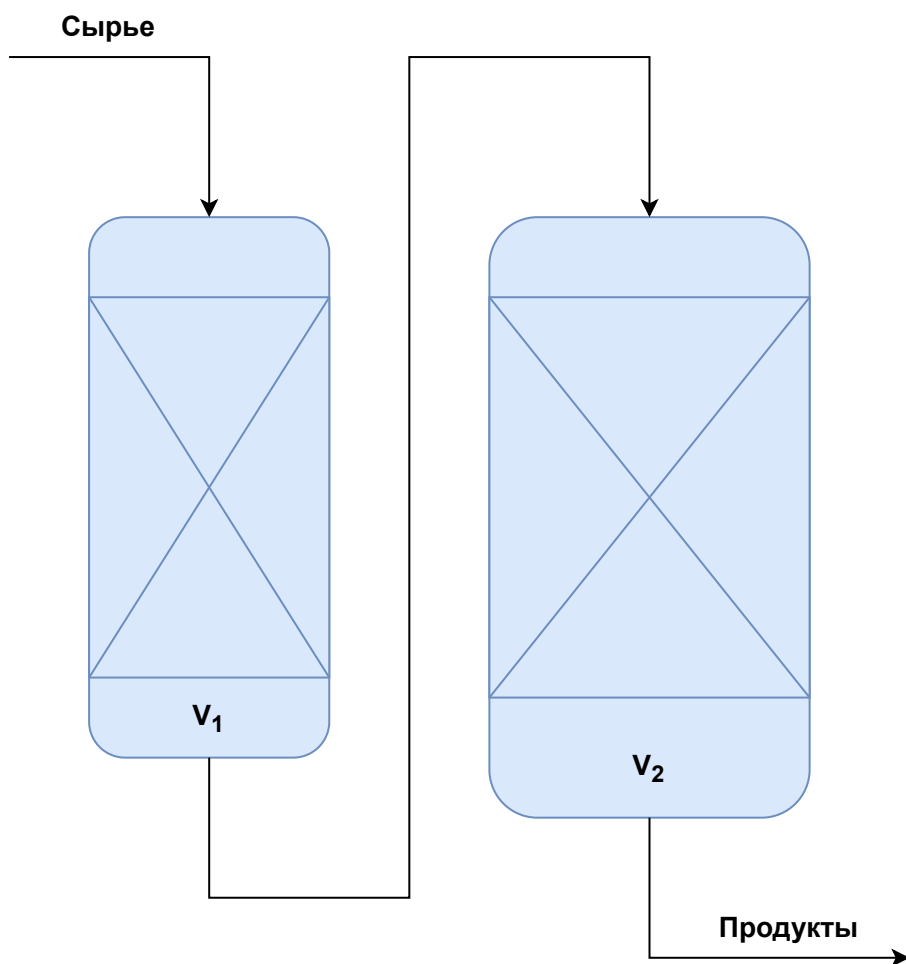
names = 'cC6H12', 'nC6H12', 'C6H6', 'H2'
molar_fractions = np.array([.8, .2, .0, .0]) # моль/л
volume_flow_rate = 10 # л/с
v = 15 # объем реактора, л
k = np.array([.4, .05, .25])

if __name__ == '__main__':
    feedstock = flow.Flow(volume_flow_rate=volume_flow_rate,
                           molar_fractions=molar_fractions)
    r1 = reactor.Reactor(v)
```

```
r1.calculate(kinetic_equations=kinetic.equations,  
            feedstock=feedstock, args=(k, ))  
products = r1.products  
  
for name, mf in zip(names, products.molar_fractions):  
    print(name, mf)  
  
r1.draw_profile(labels=names, filename='plot1.png')
```

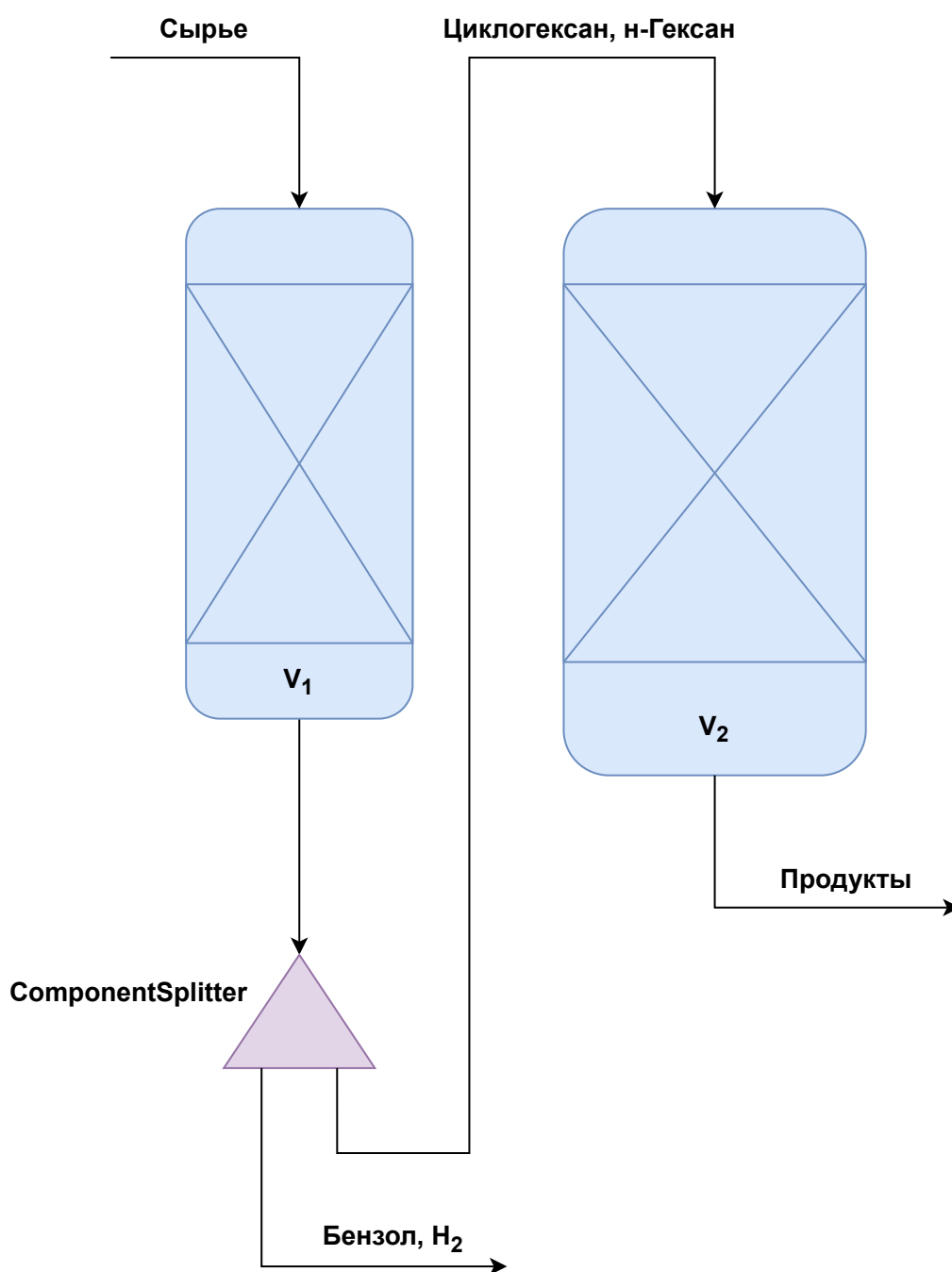


Задание 1



Необходимо рассчитать состав продуктов. Исходные данные возьмите из примера. Объем реактора $V_1 = 15$ л, объем реактора $V_2 = 1.85 \cdot V_1$.

Задание 2



1. Дополнить описание материального потока полями "массовый расход" и массовый состав (или мольный расход и состав).
2. Добавить класс `ComponentSplitter`, задача которого заключается в извлечении из состава материального потока определенных компонентов с заданной степенью четкости (от 0 до 1). При этом извлечение компонентов из состава потока влияет на его расход.

Примечание: свойства индивидуальных компонентов, входящих в состав материального потока, можно взять из базы свойств, используемой в ЛБ-05 по библиотеке Pandas.

3. Рассчитать схему реакторного блока с учетом полного извлечения из продуктов первого реактора бензола и водорода. Сравнить результаты с двухреакторной схемой из предыдущего задания.