Модули проекта

Модуль с реализацией функции, содержащей правые части системы дифференциальных уравнений (`kinetic.py`):

```python
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__':
    ...
```

Реализация класса `Flow` (`flow.py`):

```python
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`):

```python
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__':
    ...
```

Модуль основной программы (`main.py`):

```python
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')
```