МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ПРОГРАММНЫЙ КОД

Системный анализ процессов химической технологии
Лабораторная работа №8
«Идентификация кинетических параметров»

Томск – 2023 г.

Листинги

Листинг 1 – Описание кинетики с помощью матрицы (файл `kinetic.py`)

```python
import numpy as np
from scipy.integrate import solve_ivp


kinetic_matrix = np.array(
    [
        [-1, 1, 0, 1],
        [0, -1, 1, 1]
    ]
)


def kinetic_scheme(
    time: float,
    c: np.ndarray,
    k: np.ndarray,
    stoich_matrix: np.ndarray = kinetic_matrix
) -> np.ndarray:

    mask = stoich_matrix < 0
    p = (c ** -(stoich_matrix * mask)).prod(axis=1)
    reaction_rates = p * k

    right_parts = (
        (stoich_matrix.T * reaction_rates).sum(axis=1)
    )

    return right_parts


def main() -> None:
    '''
        Функция для тестирования и проверки
        работоспособности кода.
        Выводит значения правых частей
        дифференциальных уравнений.
    '''
    c = np.array([1, 0, 0, 0])
    k = np.array([.85, .5])
    time = .1
    print(kinetic_scheme(time, c, k))
    return


def test_kinetic_constants() -> None:
    """
        Функция для проверки констант скоростей.
        Выводит значения концентраций компонентов
        в каждый момент времени.
    """
    time = np.arange(0, 1.1, .1)
    k = np.array([1.7939239513204417, 1.0262965367591952])
    solution = solve_ivp(
        fun=kinetic_scheme,
        t_span=(time[0], time[-1]),
        y0=[1, 0, 0, 0],
        dense_output=True,
        args=(k, )
    )
    c9h20, c9h18, c9h16, h2  = solution.sol(time)
    for line in zip(c9h20, c9h18, c9h16, h2):
        print('{:>8.4f} {:>8.4f} {:>8.4f} {:>8.4f}'.format(*line))

    return


```

```
67 if __name__ == '__main__':
68     #main()
69     test_kinetic_constants()
```

Листинг 2 – Подбор кинетических параметров с помощью методов оптимизации функций нескольких переменных (файл `main.py`)

```python
import numpy as np
from scipy.optimize import minimize
from scipy.integrate import solve_ivp
from kinetic import kinetic_scheme
import genetic_algorithm as ga


def obj_func(
    k: np.ndarray, c: np.ndarray,
    kinetic_scheme: callable,
    time: np.ndarray,
    c0: np.ndarray,
) -> float:
    """
        Целевая функция для минимизации
    """
    solution = solve_ivp(
        fun=kinetic_scheme,
        t_span=(0, time[-1]),
        y0=c0,
        dense_output=True,
        args=(k, )
    )
    c_calc = solution.sol(time)
    return ((c - c_calc[:, 1:].T) ** 2).sum()


if __name__ == '__main__':
    data = np.loadtxt('data.txt')
    h = .1
    c0 = np.array([1, 0, 0, 0])
    time = np.arange(0, 1+h, h)

    #С использованием метода Нелдера-Мида
    res_nm = minimize(
        obj_func,
        x0=[.5, .5],
        args=(data, kinetic_scheme, time, c0),
        method='Nelder-Mead'
    )
    print(f'НелдерМид-: fun={res_nm.fun} x={res_nm.x}')

    #С использованием генетического алгоритма
    bounds = (.5, 2.), (.5, 2.)
    res_ga = ga.genetic_algorithm(
        bounds=bounds,
        func=obj_func,
        popsize=100,
        selection_size=20,
        generations_count=10,
        args=(data, kinetic_scheme, time, c0)
    )[0]
    fun = obj_func(res_ga, data, kinetic_scheme, time, c0)
    print(f'Генетический алгоритм: fun={fun} x={res_ga}')
```

Листинг 3 – Исходные данные (файл `data.txt`)

```
0.8353   0.1563   0.0084   0.1732
0.6977   0.2715   0.0308   0.3331
0.5827   0.354    0.0633   0.4805
0.4868   0.4104   0.1028   0.6161
0.4066   0.4463   0.1471   0.7405
0.3396   0.4662   0.1942   0.8546
0.2837   0.4736   0.2427   0.9591
0.2369   0.4716   0.2915   1.0545
0.1979   0.4625   0.3396   1.1417
0.1653   0.4481   0.3866   1.2213
```