TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа природных ресурсов
Направление подготовки Химическая технология
Отделение химической инженерии

PYTHON ДЛЯ ЗАДАЧ ХИМИЧЕСКОЙ ТЕХНОЛОГИИ

Отчет по лабораторной работе № 3

Выполнил студент гр. 9дм21 _____ Шуриков М К
(Подпись)

_____ _____ 2023 г.

Отчет принят:

Преподаватель
доцент ОХИ ИШПР, к.т.н. _____ В.А. Чузлов
(Подпись)

_____ _____ 2023 г.

Томск 2023 г.

Задание 1

Дана зависимость давления паров вещества от температуры:

T,°C	p, атм
40	0.2453
50	0.5459
60	1.2151
70	2.7042
80	6.0184
90	13.3943
100	29.8096

Определить значения давления паров при $T \in [40; 100]$ с шагом 5 °C, используя:

- Кубический сплайн;
- Одну из аппроксимирующих функций: проверить линейную, степенную и экспоненциальную аппроксимирующие функции, выбрать наиболее подходящую (по значению суммарной ошибки) и провести расчеты с использованием данной функции.

Программная реализация:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import interpolate

T = [float(t) for t in range(40,101) if t%10 == 0]
P = [0.2453, 0.5459, 1.2151, 2.7042, 6.0184, 13.3943, 29.8096]

def f(x):
    tck = interpolate.splrep(T, P)
    return interpolate.splev(x, tck)

Tx = [float(t) for t in range(40,101) if t%5 == 0]

print('')
print('Cubic spline')
print('T, °C P, Pa')
for t in Tx:
    print(f'{t:5.0f} {f(t):3.4f}')
print('')

from typing import Callable
from scipy.optimize import least_squares
```

```python
def linear(x: float | np.ndarray,
           params: tuple[float, float]) -> float | np.ndarray:
      a0, a1 = params
      return a0 + a1 * x

def power(x: float | np.ndarray,
          params: tuple[float, float]) -> float | np.ndarray:
      a, b = params
      return a * x ** b

def exponent(x: float | np.ndarray,
             params: tuple[float, float]) -> float | np.ndarray:
      a, b = params
      return a * np.exp(b * x)

def residuals(params: tuple[float, float], x: np.ndarray,
              y: np.ndarray, func: Callable) -> np.ndarray:
    return y - func(x, params)

#уточнить че это за параметры ниже
x0 = 0.01, 0.01

Tarray = np.array(T)
Parray = np.array(P)

print('linear')

results = least_squares(residuals, x0=x0, args=(Tarray, Parray, linear))
linear_params, linear_cost = results.x, results.cost
print(linear_params, linear_cost)

print('linear approx')
print('T, °C P, Pa')
for t in Tx:
    print(f'{t:5.0f} {linear(t, linear_params):3.4f}')

print('')
print('power')

results = least_squares(residuals, x0=x0, args=(Tarray, Parray, power))
power_params, power_cost = results.x, results.cost
print(power_params, power_cost)

print('power approx')
print('T, °C P, Pa')
for t in Tx:
    print(f'{t:5.0f} {power(t, power_params):3.4f}')

print('')
print('exponent')
```

```
results = least_squares(residuals, x0=x0, args=(Tarray, Parray, exponent))
exponent_params, exponent_cost = results.x, results.cost
print(exponent_params, exponent_cost)

print('exponent approx')
print('T, °C P, Pa')
for t in Tx:
    print(f'{t:5.0f} {exponent(t, exponent_params):3.4f}')
print('')
```

Ответ:

```
Cubic spline
T, °C P, Pa
   40 0.2453
   45 0.3709
   50 0.5459
   55 0.8131
   60 1.2151
   65 1.8085
   70 2.7042
   75 4.0417
   80 6.0184
   85 8.9285
   90 13.3943
   95 20.1199
  100 29.8096

linear
[-22.09356439   0.42568929] 95.2304518735714
linear approx
T, °C P, Pa
   40 -5.0660
   45 -2.9375
   50 -0.8091
   55 1.3193
   60 3.4478
   65 5.5762
   70 7.7047
   75 9.8331
   80 11.9616
   85 14.0900
   90 16.2185
   95 18.3469
  100 20.4754

power
[1.31403122e-11 6.16745105e+00] 2.804176924307334
power approx
```

```
T, °C P, Pa
   40 0.0998
   45 0.2064
   50 0.3953
   55 0.7116
   60 1.2169
   65 1.9937
   70 3.1489
   75 4.8190
   80 7.1750
   85 10.4280
   90 14.8354
   95 20.7071
  100 28.4124

exponent
[0.00999958 0.08000043] 4.008900711405008e-09
exponent approx
T, °C P, Pa
   40 0.2453
   45 0.3660
   50 0.5460
   55 0.8145
   60 1.2151
   65 1.8127
   70 2.7042
   75 4.0342
   80 6.0184
   85 8.9784
   90 13.3943
   95 19.9819
  100 29.8096
```

Задание 2

Дана схема химических превращений:

$$A \overset{k_1}{\leftarrow} B \underset{k_3}{\overset{k_2}{\leftrightarrow}} C$$

$C_{A_0} = 0.8$ моль/л; $k_1 = 0.8\ (c^{-1})$

$C_{B_0} = 0.2$ моль/л; $k_2 = 0.96\ (c^{-1})$

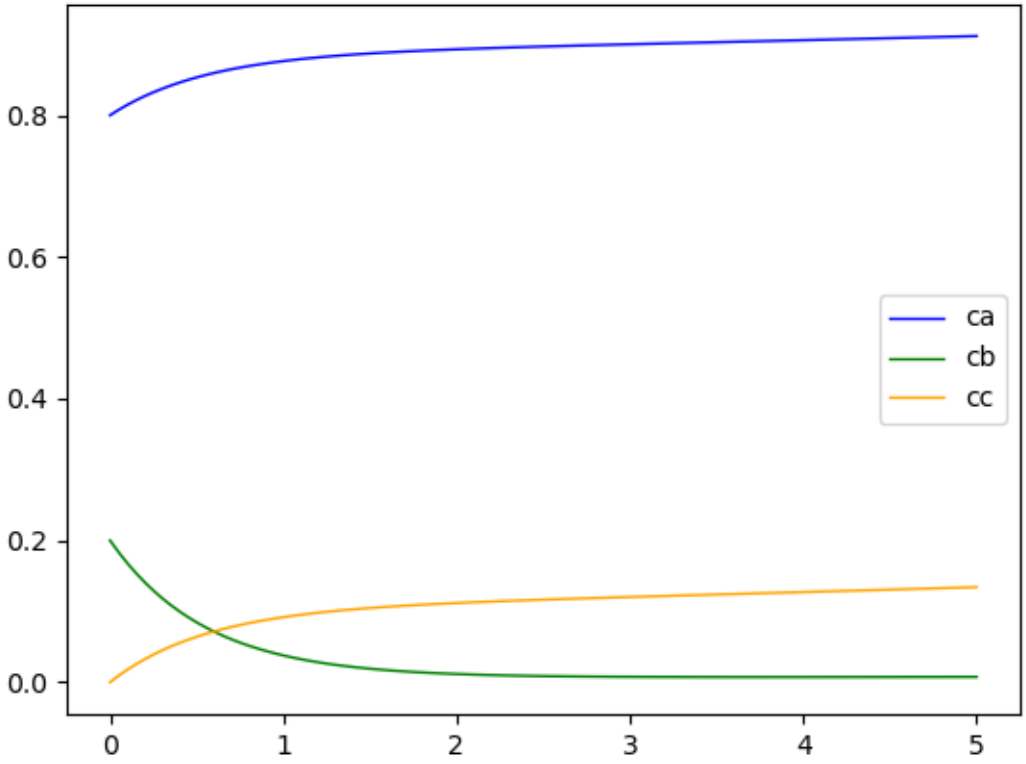
$C_{C_0} = 0.0$ моль/л; $k_3 = 0.1\ (c^{-1})$

Решите систему дифференциальных уравнений изменения концентраций веществ во времени при помощи функции `scipy.integrate.solve_ivp()` на отрезке [0; 5] с шагом h = 0.1. По результатам расчетов постройте зависимость C(t) для каждого компонента при помощи библиотеки `matplotlib`.

Программная реализация:

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

# нач концентрация, mol/l
ca0 = 0.8
cb0 = 0.2
cc0 = 0.0

# конст скорости, sec-1
k1 = 0.8
k2 = 0.96
k3 = 0.1

# дифуры
def func(t, c):
    ca, cb, cc = c
    dcadt = k1 * cb
    dcbdt = - k1 * cb - k2 * cb + k3 * cc
    dccdt = k2 * cb
    return dcadt, dcbdt, dccdt

# итегрирование
t = np.linspace(0, 5, 100, endpoint = 5)
solution = solve_ivp(func, (0, 5), [ca0, cb0, cc0], dense_output = True)
ca = solution.sol(t)[0]
cb = solution.sol(t)[1]
cc = solution.sol(t)[2]

# построение кинетических кривых
fig, ax = plt.subplots()

ax.plot(t,ca, 'b', linewidth = 1, label = 'ca')
ax.plot(t,cb, 'g', linewidth = 1, label = 'cb')
ax.plot(t,cc, 'orange', linewidth = 1, label = 'cc')

plt.legend(loc='best')
plt.show()
```

Ответ:



Задание 3

Используя функцию `scipy.integrate.quad()` для вычисления значения энтропии воды при ее нагревании от 400 до 500 K по формуле:

$$\Delta S = \eta \int_{400}^{500} \frac{C_v(T)dT}{T}$$

$$C_v(T) = R \sum_{j=1}^{12} A_j \tau^{j-1}$$

$$\tau = 1 - \frac{T}{T_c}$$

где T — температура, К; η = 3 — количество молей; Cv — теплоемкость, Дж/(моль К); R — универсальная газовая постоянная; Tc = 647.126 — критическая температура, К.

Коэффициенты полинома A(1) − A(12):

Коэффициент	Значение
A_1	7.4305055
A_2	-24.93618016
A_3	195.5654567
A_4	1986.485797
A_5	-53305.43411
A_6	505697.1723
A_7	-2724774.677
A_8	9167737.673
A_9	-19622033.78
A_{10}	25984725.33
A_{11}	-19419431.35
A_{12}	6263206.554

Программная реализация:

```
import numpy as np
from numpy.polynomial import Polynomial
from scipy.integrate import quad

n = 3
R = 8.31
Tc = 647.126

T = np.linspace(400, 500, 1000, endpoint = 500)

# thau
def thau(
        T: float,
) -> np.ndarray:
    thau = 1 - T/Tc
    return thau

#теплоемкость
def Cv_func(
        T: float,
        R: float,
)->list[float]:
    thu = thau(T)
    p = Polynomial([6263206.554, -19419431.35, 25984725.33, -19622033.78,
9167737.673, -2724774.677, 505697.1723, -53305.43411, 1986.485797, 195.5654567, -
24.93618016, 7.4305055][::-1])
    Cv = R * p(thu)
```

```
    devide = Cv / T
    return devide

I = quad(Cv_func, 400, 500, args=(R, ))

multiplication = I[0] * 3

print (f'{multiplication:0.2f}')
```

Ответ:

41.29