Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Российский химико-технологический университет имени Д.И. Менделеева»

Факультет цифровых технологий и химического инжиниринга

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 6

ПО КУРСУ

«ЦИФРОВОЕ МОДЕЛИРОВАНИЕ ФИЗИКО-ХИМИЧЕСКИХ СИСТЕМ»:

«Определение порядка, скорости реакции. Расчет константы скорости реакции. Расчет текущих концентраций для сложных реакций»

Ведущий преподаватель

Ст. преподаватель Скичко Е.А.

СТУДЕНТ группы КС-26 Золотухин А.А.

Москва

2024

Задание

1. Дана зависимость общего давления идеальной газовой смеси p от времени t при постоянной температуре T в ходе реакции A ($2Cl_2O \rightarrow 2Cl_2 + O_2$) (*Табл. 1*).

Таблица 1

Вариант 7	$2Cl_2O \rightarrow 2Cl_2 + O_2$		
T, К	t, с	p, кПа	t_1, с
458	0	4, 21	250
	30	4, 51	
	60	4, 73	
	120	5, 04	
	240	5, 40	
	360	5, 52	

По приведенным данным рассчитайте парциальное давление и концентрацию реагента в соответствующие моменты времени, постройте кинетическую кривую концентрации или парциального давления реагента. Графическим и аналитическим методами подбора определите константу скорости и порядок реакции A. Рассчитайте время полупревращения исходного вещества в данном опыте, а также его концентрацию и степень превращения в момент времени t_1 после начала реакции. В момент начала опыта в системе присутствовал только исходный реагент.

2. Порядок выполнения задания:
    a. В соответствии с заданием (*Табл. 2*) составить математическое описание химического реактора.
    b. Разработать алгоритм и программу расчета.
    c. Провести расчеты изменения концентраций веществ от времени.
    d. Полученные результаты оформить в виде таблиц и графиков.
    e. Составить отчет о проделанной работе.

Таблица 2

№ варианта	Тип реакции	Исходная концентрация, кмоль/м^3	Константы скорости	Энергии активации, кДж/моль	Температура, К
10	$C_8H_{18} \rightarrow (k_1) i\text{-}C_8H_{18} \rightarrow (k_2)$ $C_4H_{10}+C_4H_8$	C_{C8H18}=0,036	k_1=0,12 k_2=0,80	E_1=94,2 E_2=81,2	620

Теоретическое обоснование решения

Есть реакция:

$$2Cl_2O \rightarrow 2Cl_2 \uparrow + O_2 \uparrow \qquad (1)$$

Реакция (1) есть реакция разложения оксида хлора (I) идет по **цепному механизму**[1] , т.е. является **сложной**[2]. При определении порядка и константы скорости данной реакции буду использовать методы **формальной кинетики**[3], считая реакцию **односторонней**[4].

---

[1] механизм, при котором исходные вещества вступают в цепь превращений с участием промежуточных активных частиц (*интермедиатов*) и их регенерацией в каждом *элементарном акте* (в одну стадию) реакции.
[2] многостадийная реакция.
[3] раздел химической кинетики, в котором рассматривают зависимость скорости реакции от концентраций реагирующих веществ в закрытых системах при постоянной температуре.
[4] реакции, в которых конечные продукты отсутствуют в начальный момент времени.

Воспользуюсь интегральным методом подбора уравнения и координат. Для этого потребуется вычислить значения парциального давления оксида хлора (I). Составлю материальный баланс реакции, записывая данные под соответствующей формулой реагента или продукта (*Табл. 1*):

Таблица 1

Уравнение реакции:	$2Cl_2O \rightarrow 2Cl_2\uparrow + O_2\uparrow$		
Начало опыта ($t = 0$)	p_0	0	0
Ушло к t	p_x	p_x	$\frac{1}{2}p_x$
Текущие значения ($t > 0$)	$p_0 - p_x$	p_x	$\frac{1}{2}p_x$

Общее давление p в системе согласно **закону Дальтона**[5]:

$$p = p_0 - p_x + p_x + \frac{1}{2}p_x = p_0 + \frac{1}{2}p_x$$

откуда

$$p_x = 2p - 2p_0$$

Теперь есть возможность выразить текущее парциальное давление оксида хлора (I) (обозначу его p_{Cl_2}) через общее давление:

$$p_{Cl_2} = p_0 - p_x = p_0 - (2p - 2p_0) = 3p_0 - 2p. \qquad (2)$$

[5] давление смеси идеальных газов (p, Па) равно сумме парциальных давлений (p_i, Па), входящих в нее газов (n).

Рассчитаю парциальное давление оксида хлора (I) по формуле *(2)*.

Применив **уравнение состояния идеального газа**, или уравнение Менделеева-Клапейрона:

$$p \cdot V = n \cdot R \cdot T, \quad (3)$$

где p - давление газа, Па;

V - объем газа, $м^3$;

n - количество вещества, моль;

R - универсальная газовая постоянная, Дж/(моль * К);

T - термодинамическая температура, К

выражу концентрацию газа, зная, что она численно равна:

$$C = \frac{n}{V} \quad (4)$$

Объединив *(3)* и *(4)*, получаю:

$$C_{Cl_2} = \frac{p_A}{RT}$$

Чтобы найти константу равновесия, можно вспомнить **основной постулат химической кинетики** (закон действующих масс Гульдберга-Вааге): скорость реакции в каждый момент времени пропорциональна произведению концентраций (C_i, моль/$м^3$) реагирующих веществ, возведенных в некоторые степени (n_i):

$$r = k \prod_{i=1}^{n} C_i^{n_i}, \quad (5)$$

где k - константа скорости, $(моль/м^3)^{(1-n)}/с$.

С другой стороны, мгновенная скорость (r, моль/(м^3*с)) реакции может быть определена также как производная от концентрации по времени, т.е.:

$$r = \pm \frac{1}{V}\frac{dn_i}{dt} = \frac{dC_i}{dt}, \quad (6)$$

где V - объем системы;

n_i - количество i-го вещества, моль,

$\frac{dn_i}{dt}$ - производная от количества вещества по времени, которая положительная для продуктов и отрицательна для реагентов.

Перейду к такой кинетической характеристике процессов, как **инвариантная скорость**[6]. Для этого используют понятие **глубины превращения** (ξ, моль), которая определяется уравнением:

$$\xi = \frac{n_{i0} - n_{it}}{v_i}, \quad (6)$$

где n_{i0} - количество i-го вещества в начале реакции, моль;

n_{it} - количество i-го вещества в момент времени t, моль;

v_i - стехиометрический коэффициент перед i-м веществом в уравнении химической реакции, которому приписывают знак “минус” (в случае реагентов) или знак “плюс” (в случае продуктов).

С учетом *(6)* количество i-го вещества:

$$n_i = v_i d\xi. \quad (7)$$

Учитывая *(6)* и *(7)*, получаю выражение для инвариантной скорости:

[6] скорость, которую используют при процессах с известной стехиометрией и которая не зависит от выбора компонента.

$$r = \pm \frac{1}{V}\frac{d\xi}{dt} = \pm \frac{1}{v_i V}\frac{dn_i}{dt} = \pm \frac{1}{v_i}\frac{dC_i}{dt}. \quad (8)$$

Возвращаясь к *(1)* и используя *(5)* и *(8)*, найду составлю дифференциальное уравнение для **скорости реакции нулевого порядка**:

$$r = -\frac{dC_{Cl_2}}{dt} = kC^0_{Cl_2} = k.$$

Умножив обе части на *dt*, проинтегрирую в пределах от $C^0_{Cl_2}$ (начальная концентрация) до C_{Cl_2} (текущая концентрация) и от *0* до *t*:

$$-\int_{C^0_{Cl_2}}^{C_{Cl_2}} dC_{Cl_2} = \int_0^t kdt,$$

получаю интегральную форму кинетического уравнения:

$$C_{Cl_2} - C^0_{Cl_2} = -kt. \quad (9)$$

Из *(9)* можно выразить текущую концентрацию:

$$C_{Cl_2} = C^0_{Cl_2} - kt \quad (10)$$

и константу скорости:

$$k = \frac{C^0_{Cl_2} - C_{Cl_2}}{t} \quad (11)$$

Время полупревращения[7] необратимой реакции нулевого порядка, исходя из *(11)*:

$$t = \frac{C^0_{Cl_2} - C_{Cl_2}}{k} = \frac{C^0_{Cl_2} - \frac{1}{2}C^0_{Cl_2}}{k} = \frac{C^0_{Cl_2}}{2k}.$$

Возвращаясь к *(1)* и используя *(5)* и *(8)*, найду составлю дифференциальное уравнение для **скорости реакции первого порядка**:

$$r = -\frac{dC_{Cl_2}}{dt} = kC^1_{Cl_2} = kC_{Cl_2}.$$

Умножив обе части на *dt* и разделив на C_{Cl_2}, проинтегрирую в пределах от $C^0_{Cl_2}$ (начальная концентрация) до C_{Cl_2}(текущая концентрация) и от *0* до *t*:

$$-\int_{C^0_{Cl_2}}^{C_{Cl_2}} \frac{dC_{Cl_2}}{C_{Cl_2}} = \int_0^t kdt,$$

получаю интегральную форму кинетического уравнения:

$$lnC_{Cl_2} - lnC^0_{Cl_2} = -kt. \qquad (12)$$

Из *(9)* можно выразить текущую концентрацию:

$$C_{Cl_2} = C^0_{Cl_2} e^{-kt} \qquad (13)$$

и константу скорости:

[7] время, за которое исходная концентрация уменьшится в 2 раза.

$$k = \frac{1}{t} ln \frac{C^0_{Cl_2}}{C_{Cl_2}}. \quad (14)$$

Время полупревращения необратимой реакции первого порядка, исходя из *(14)*:

$$t = \frac{1}{k} ln \frac{C^0_{Cl_2}}{C_{Cl_2}} = \frac{1}{k} ln \frac{C^0_{Cl_2}}{\frac{1}{2} C^0_{Cl_2}} = \frac{ln2}{k}.$$

Возвращаясь к *(1)* и используя *(5)* и *(8)*, найду составлю дифференциальное уравнение для **скорости реакции второго порядка**:

$$r = - \frac{dC_{Cl_2}}{dt} = kC^2_{Cl_2}.$$

Умножив обе части на *dt* и разделив на $C^2_{Cl_2}$, проинтегрирую в пределах от $C^0_{Cl_2}$ (начальная концентрация) до C_{Cl_2} (текущая концентрация) и от *0* до *t*:

$$- \int_{C^0_{Cl_2}}^{C_{Cl_2}} \frac{dC_{Cl_2}}{C^2_{Cl_2}} = \int_0^t kdt,$$

получаю интегральную форму кинетического уравнения:

$$\frac{1}{C_{Cl_2}} - \frac{1}{C^0_{Cl_2}} = kt. \quad (15)$$

Из *(9)* можно выразить текущую концентрацию:

$$C_{Cl_2} = \frac{1}{kt + \frac{1}{C^0_{Cl_2}}} \quad (16)$$

и константу скорости:

$$k = \frac{1}{t}\left(\frac{1}{C_{Cl_2}} - \frac{1}{C^0_{Cl_2}}\right) \quad (17)$$

Время полупревращения необратимой реакции второго порядка, исходя из *(17)*:

$$t = \frac{1}{k}\left(\frac{1}{C^0_{Cl_2}} - \frac{1}{\frac{1}{2}C^0_{Cl_2}}\right) = \frac{1}{kC^0_{Cl_2}}$$

Найду **степень превращения** в момент времени t_1 после начала реакции как отношение убыли концентрации реагента Cl_2 в ходе реакции ($C^0_{Cl_2} - C_{Cl_2}$, моль/м^3) к исходной концентрации ($C^0_{Cl_2}$, моль/м^3):

$$\alpha = \frac{C^0_{Cl_2} - C_{Cl_2}}{C^0_{Cl_2}}.$$

Есть необратимая последовательная реакция:

$$C_8H_{18} \rightarrow (k_1)\text{i-}C_8H_{18} \rightarrow (k_2)C_4H_{10} + C_4H_8 \quad (18)$$

Принцип независимости элементарных реакций утверждает, что константа скорости элементарной химической реакции не зависит от того, протекают ли в данной системе одновременно другие элементарные реакции. При справедливости этого положения скорость реакции по компоненту равна алгебраической сумме скоростей его образования и расходования во всех стадиях, т.е.:

$$r = \frac{dC_i}{dt} = \sum_{i=1}^{x} v_{i,s} r_s, \quad (19)$$

где r_s - скорость отдельной стадии, моль / (м^3 * с);

v_i - стехиометрический коэффициент перед компонентом в уравнении реакции для *i*-ой стадии сложного процесса.

Исходя из *(18)* и *(19)*, получаю систему дифференциальных уравнений:

$$-\frac{dC_{C_8H_{18}}}{dt} = r_1 = k_1 C_{C_8H_{18}}, \quad (19)$$

$$\frac{dC_{i-C_8H_{18}}}{dt} = -r_2 + r_1 = k_1 C_{C_8H_{18}} - k_2 C_{i-C_8H_{18}}, \quad (20)$$

$$\frac{dC_{C_4H_{10}}}{dt} = r_2 = k_2 C_{i-C_8H_{18}}, \quad (21)$$

$$\frac{dC_{C_4H_8}}{dt} = r_2 = k_2 C_{i-C_8H_{18}}. \quad (22)$$

Систему из дифференциальных уравнений *(19)-(22)* буду решать с помощью **метода Рунге-Кутты 7-го порядка**. Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка (далее $y,\ f,\ k_i \in R^n,\ h \in R^1$):

$$y' = f(x, y),\ y(x_0) = y_0.$$

Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:

$$y_{n+1} = y_n + \frac{41k_1+216k_4+27k_5+272k_6+27k_7+216k_8+41k_9}{840}.$$

Метод седьмого порядка должен иметь по меньшей мере девять стадий:

$$k_1 = hf(x_n, y_n),$$

$$k_2 = hf(x_n + \frac{h}{12}, y_n + \frac{k_1}{12}),$$

$$k_3 = hf(x_n + \frac{h}{12}, y_n + \frac{-10k_1+11k_2}{12}),$$

$$k_4 = hf(x_n + \frac{2h}{12}, y_n + \frac{12k_3}{12}),$$

$$k_5 = hf(x_n + \frac{4h}{12}, y_n + \frac{157k_1-318k_2+4k_3+160k_4}{9}),$$

$$k_6 = hf(x_n + \frac{6h}{12}, y_n + \frac{-322k_1+199k_2+108k_3-131k_5}{30}),$$

$$k_7 = hf(x_n + \frac{8h}{12}, y_n + \frac{3158k_1}{45} - \frac{638k_2}{6} - \frac{23k_3}{2} + \frac{157k_4}{3} + \frac{157k_6}{45}),$$

$$k_8 = hf(x_n + \frac{10h}{12}, y_n - \frac{53k_1}{14} + \frac{38k_2}{7} - \frac{3k_3}{14} - \frac{65k_4}{72} + \frac{29k_7}{90}),$$

$$k_9 = hf(x_n + h, y_n + \frac{56k_1}{25} + \frac{283k_2}{14} - \frac{119k_3}{6} - \frac{26k_4}{7} - \frac{13k_5}{15} + \frac{149k_6}{32} - \frac{25k_7}{9} + \frac{27k_8}{25}).$$

Уравнение Аррениуса:

$$k = A \cdot e^{\frac{-E_a}{RT}},$$

где *k* - константа скорости реакции;

e - основание натурального логарифма;

T - термодинамическая температура, К;

R - универсальная газовая постоянная, Дж/(моль * К);

E_a - энергия активации, Дж/моль;

A - предэкспоненциальный множитель, показывающий общее число столкновений.

Код

```rust
use std::{fs::File, io::prelude::*};


// объявление структуры исходных данных
#[derive(Debug)]
struct InitialData {
    universal_gas_equation_system: f32, // универсальная газовая постоянная
    temperature_1: f32, // температура газовой смеси (задание 1)
    time: Vec<i32>, // время (задание 1)
    pressure: Vec<f32>, // давление газовой смеси (задание 1)
    time_1: i32, // момент времени после начала реакции (задание 1)
    temperature_2: f32, // температура (задание 2)
    k_1: f32, // константа скорости первой реакции (задание 2)
    k_2: f32, // константа скорости второй реакции (задание 2)
    c_c8h18: f32, // начальная концентрация вещества C8H18 (задание 2)
    c_ic8h18: f32, // начальная концентрация вещества i-C8H18 (задание 2)
    c_c4h10: f32, // начальная концентрация вещества C4H10 (задание 2)
    c_c4h8: f32, // начальная концентрация вещества C4H8 (задание 2)
    dt: f32, // шаг в методе Рунге-Кутты 4-го порядка (задание 2)
    number_of_iterations: Vec<usize>, // количество итераций
}


impl InitialData {
```

```
    // вычисление давления вещества A
    fn pressure_calc(&self) -> Vec<f32> {
        let mut temporary: Vec<f32> = vec![];
        for i in 0..self.pressure.len() {
            temporary.push(self.pressure[0] - (2.0 * self.pressure[i] - 2.0 *
self.pressure[0]));
        }
        return temporary;
    }

    // вычисление концентрации вещества A
    fn concentration_calc(&self, pressure_a: Vec<f32>) -> Vec<f32> {
        let mut temporary: Vec<f32> = vec![];
        for i in 0..pressure_a.len() {
            temporary.push(pressure_a[i] / (self.temperature_1 *
self.universal_gas_equation_system));
        }
        return temporary;
    }

    // вычисление анаморфозы нулевого порядка
    fn anamorphosis0(&self, concentration_a: Vec<f32>) -> Vec<f32> {
        let mut temporary: Vec<f32> = vec![];
        for i in 0..concentration_a.len() {
            temporary.push(concentration_a[0] - concentration_a[i]);
        }
        return temporary;
    }
```

```rust
    // вычисление анаморфозы первого порядка
    fn anamorphosis1(&self, concentration_a: Vec<f32>) -> Vec<f32> {
        let mut temporary: Vec<f32> = vec![];
        for i in 0..concentration_a.len() {
            temporary.push((concentration_a[0] / concentration_a[i]).ln());
        }
        return temporary;
    }

    // вычисление анаморфозы второго порядка
    fn anamorphosis2(&self, concentration_a: Vec<f32>) -> Vec<f32> {
        let mut temporary: Vec<f32> = vec![];
        for i in 0..concentration_a.len() {
            temporary.push(1.0 / concentration_a[i] - 1.0 / concentration_a[0]);
        }
        return temporary;
    }

    // вычисление константы скорости аналитическим методом для 0-го порядка
    fn rate_constant0_calc(&self, concentration_a: Vec<f32>) -> Vec<f32> {
        let mut temporary: Vec<f32> = vec![];
        for i in 0..concentration_a.len() {
            temporary.push((1 as f32 / self.time[i] as f32) * (concentration_a[0] -
concentration_a[i]));
        }
        return temporary;
```

```
    }


    // вычисление константы скорости аналитическим методом для 1-го порядка
    fn rate_constant1_calc(&self, concentration_a: Vec<f32>) -> Vec<f32> {
        let mut temporary: Vec<f32> = vec![];
        for i in 0..concentration_a.len() {
            temporary.push((1 as f32 / self.time[i] as f32) * (concentration_a[0] / concentration_a[i]).ln());
        }
        return temporary;
    }


    // вычисление константы скорости аналитическим методом для 2-го порядка
    fn rate_constant2_calc(&self, concentration_a: Vec<f32>) -> Vec<f32> {
        let mut temporary: Vec<f32> = vec![];
        for i in 0..concentration_a.len() {
            temporary.push((1 as f32 / self.time[i] as f32) * (1.0 / concentration_a[i] - 1.0 / concentration_a[0]));
        }
        return temporary;
    }


    // вычисление времени полупревращения, концентрации и степени превращения в момент времени t1
    fn time_moment(&self, moment: f32, rate_constant: f32, concentration0: f32) -> (f32, f32, f32) {
        let mut half_turn_time: f32 = 0.0;
        let mut concentration_time_moment: f32 = 0.0;
        let mut tranformation_degree: f32 = 0.0;
```

```
        half_turn_time = 1.0 / (rate_constant * concentration0);
        concentration_time_moment = 1.0 / (rate_constant * moment + 1.0 /
concentration0);
        tranformation_degree = rate_constant * moment *
concentration_time_moment;
        return (half_turn_time, concentration_time_moment,
tranformation_degree);
    }



    // решение системы дифференциальных уравнений методом Рунге-Кутты 7-го порядка
    fn equation_system(&self, mut c_c8h18_mod: Vec<f32>, mut c_ic8h18_mod: Vec<f32>,
mut c_c4h10_mod: Vec<f32>, mut c_c4h8_mod: Vec<f32>) -> (Vec<f32>, Vec<f32>,
Vec<f32>, Vec<f32>) {
        for i in 0..self.number_of_iterations.len() {
            let h = self.dt;
            let k1_c8h18 = -self.k_1 * c_c8h18_mod[i];
            let k1_ic8h18 = self.k_1 * c_c8h18_mod[i] - self.k_2 * c_ic8h18_mod[i];
            let k1_c4h10 = self.k_2 * c_ic8h18_mod[i];
            let k1_c4h8 = self.k_2 * c_ic8h18_mod[i];

            let k2_c8h18 = -self.k_1 * (c_c8h18_mod[i] + h * (1 as f32 / 12 as f32) *
k1_c8h18);
            let k2_ic8h18 = self.k_1 * (c_c8h18_mod[i] + h * (1 as f32 / 12 as f32) *
k1_c8h18) - self.k_2 * (c_ic8h18_mod[i] + h * (1 as f32 / 12 as f32) * k1_ic8h18);
            let k2_c4h10 = self.k_2 * (c_ic8h18_mod[i] + h * (1 as f32 / 12 as f32) *
k1_ic8h18);
            let k2_c4h8 = self.k_2 * (c_ic8h18_mod[i] + h * (1 as f32 / 12 as f32) *
k1_ic8h18);
```

```
            let k3_c8h18 = -self.k_1 * (c_c8h18_mod[i] + h * (-10.0 * k1_c8h18 + 11.0 * k2_c8h18)/12.0);

            let k3_ic8h18 = self.k_1 * (c_c8h18_mod[i] + h * (-10.0 * k1_c8h18 + 11.0 * k2_c8h18)/12.0) - self.k_2 * (c_ic8h18_mod[i] + h * (-10.0 * k1_ic8h18 + 11.0 * k2_ic8h18)/12.0);

            let k3_c4h10 = self.k_2 * (c_ic8h18_mod[i] + h * (-10.0 * k1_ic8h18 + 11.0 * k2_ic8h18)/12.0);

            let k3_c4h8 = self.k_2 * (c_ic8h18_mod[i] + h * (-10.0 * k1_ic8h18 + 11.0 * k2_ic8h18)/12.0);


            let k4_c8h18 = -self.k_1 * (c_c8h18_mod[i] + h * (2 as f32 / 12 as f32) * k3_c8h18);

            let k4_ic8h18 = self.k_1 * (c_c8h18_mod[i] + h * (2 as f32 / 12 as f32) * k3_c8h18) - self.k_2 * (c_ic8h18_mod[i] + h * (2 as f32 / 12 as f32) * k3_ic8h18);

            let k4_c4h10 = self.k_2 * (c_ic8h18_mod[i] + h * (2 as f32 / 12 as f32) * k3_ic8h18);

            let k4_c4h8 = self.k_2 * (c_ic8h18_mod[i] + h * (2 as f32 / 12 as f32) * k3_ic8h18);


            let k5_c8h18 = -self.k_1 * (c_c8h18_mod[i] + h * (157.0 * k1_c8h18 - 318.0 * k1_c8h18 + 4.0 * k1_c8h18 + 160.0 * k1_c8h18)/9.0);

            let k5_ic8h18 = self.k_1 * (c_c8h18_mod[i] + h * (157.0 * k1_c8h18 - 318.0 * k1_c8h18 + 4.0 * k1_c8h18 + 160.0 * k1_c8h18)/9.0) - self.k_2 * (c_ic8h18_mod[i] + h * (157.0 * k1_ic8h18 - 318.0 * k1_ic8h18 + 4.0 * k1_ic8h18 + 160.0 * k1_ic8h18)/9.0);

            let k5_c4h10 = self.k_2 * (c_ic8h18_mod[i] + h * (157.0 * k1_ic8h18 - 318.0 * k1_ic8h18 + 4.0 * k1_ic8h18 + 160.0 * k1_ic8h18)/9.0);

            let k5_c4h8 = self.k_2 * (c_ic8h18_mod[i] + h * (157.0 * k1_ic8h18 - 318.0 * k1_ic8h18 + 4.0 * k1_ic8h18 + 160.0 * k1_ic8h18)/9.0);


            let k6_c8h18 = (-self.k_1 * (c_c8h18_mod[i] + h * (-322.0 * k1_c8h18 + 199.0 * k1_c8h18 + 108.0 * k1_c8h18 - 131.0 * k1_c8h18)/30.0)) as f32;
```

```
            let k6_ic8h18 = self.k_1 * (c_c8h18_mod[i] + h * (-322.0 * k1_c8h18 +
199.0 * k1_c8h18 + 108.0 * k1_c8h18 - 131.0 * k1_c8h18)/30.0) - self.k_2 *
(c_ic8h18_mod[i] + h * (-322.0 * k1_ic8h18 + 199.0 * k1_ic8h18 + 108.0 * k1_ic8h18 -
131.0 * k1_ic8h18)/30.0);

            let k6_c4h10 = self.k_2 * (c_ic8h18_mod[i] + h * (-322.0 * k1_ic8h18 +
199.0 * k1_ic8h18 + 108.0 * k1_ic8h18 - 131.0 * k1_ic8h18)/30.0);

            let k6_c4h8 = self.k_2 * (c_ic8h18_mod[i] + h * (-322.0 * k1_ic8h18 +
199.0 * k1_ic8h18 + 108.0 * k1_ic8h18 - 131.0 * k1_ic8h18)/30.0);


            let k7_c8h18 = -self.k_1 * (c_c8h18_mod[i] + h * ((3158 as f32 / 45 as
f32) * k1_c8h18 - (638 as f32 / 6 as f32) * k2_c8h18 - (23 as f32 / 2 as f32) *
k3_c8h18 + (157 as f32 / 3 as f32) * k4_c8h18 + (157 as f32 / 45 as f32) *
k6_c8h18));

            let k7_ic8h18 = self.k_1 * (c_c8h18_mod[i] + h * ((3158 as f32 / 45 as
f32) * k1_c8h18 - (638 as f32 / 6 as f32) * k2_c8h18 - (23 as f32 / 2 as f32) *
k3_c8h18 + (157 as f32 / 3 as f32) * k4_c8h18 + (157 as f32 / 45 as f32) * k6_c8h18))
- self.k_2 * (c_ic8h18_mod[i] + h * ((3158 as f32 / 45 as f32) * k1_ic8h18 - (638 as
f32 / 6 as f32) * k2_ic8h18 - (23 as f32 / 2 as f32) * k3_ic8h18 + (157 as f32 / 3 as
f32) * k4_ic8h18 + (157 as f32 / 45 as f32) * k6_ic8h18));

            let k7_c4h10 = self.k_2 * (c_ic8h18_mod[i] + h * ((3158 as f32 / 45 as
f32) * k1_ic8h18 - (638 as f32 / 6 as f32) * k2_ic8h18 - (23 as f32 / 2 as f32) *
k3_ic8h18 + (157 as f32 / 3 as f32) * k4_ic8h18 + (157 as f32 / 45 as f32) *
k6_ic8h18));

            let k7_c4h8 = self.k_2 * (c_ic8h18_mod[i] + h * ((3158 as f32 / 45 as
f32) * k1_ic8h18 - (638 as f32 / 6 as f32) * k2_ic8h18 - (23 as f32 / 2 as f32) *
k3_ic8h18 + (157 as f32 / 3 as f32) * k4_ic8h18 + (157 as f32 / 45 as f32) *
k6_ic8h18));


            let k8_c8h18 = -self.k_1 * (c_c8h18_mod[i] + h * (- (53 as f32 / 14 as
f32) * k1_c8h18 + (38 as f32 / 7 as f32) * k2_c8h18 - (3 as f32 / 14 as f32) *
k3_c8h18 - (65 as f32 / 72 as f32) * k5_c8h18 + (29 as f32 / 90 as f32) * k7_c8h18));

            let k8_ic8h18 = self.k_1 * (c_c8h18_mod[i] + h * (- (53 as f32 / 14 as
f32) * k1_c8h18 + (38 as f32 / 7 as f32) * k2_c8h18 - (3 as f32 / 14 as f32) *
k3_c8h18 - (65 as f32 / 72 as f32) * k5_c8h18 + (29 as f32 / 90 as f32) * k7_c8h18))
```

```
- self.k_2 * (c_ic8h18_mod[i] + h * (- (53 as f32 / 14 as f32) * k1_ic8h18 + (38 as f32 / 7 as f32) * k2_ic8h18 - (3 as f32 / 14 as f32) * k3_ic8h18 - (65 as f32 / 72 as f32) * k5_ic8h18 + (29 as f32 / 90 as f32) * k7_ic8h18));

            let k8_c4h10 = self.k_2 * (c_ic8h18_mod[i] + h * (- (53 as f32 / 14 as f32) * k1_ic8h18 + (38 as f32 / 7 as f32) * k2_ic8h18 - (3 as f32 / 14 as f32) * k3_ic8h18 - (65 as f32 / 72 as f32) * k5_ic8h18 + (29 as f32 / 90 as f32) * k7_ic8h18));

            let k8_c4h8 = self.k_2 * (c_ic8h18_mod[i] + h * (- (53 as f32 / 14 as f32) * k1_ic8h18 + (38 as f32 / 7 as f32) * k2_ic8h18 - (3 as f32 / 14 as f32) * k3_ic8h18 - (65 as f32 / 72 as f32) * k5_ic8h18 + (29 as f32 / 90 as f32) * k7_ic8h18));


            let k9_c8h18 = -self.k_1 * (c_c8h18_mod[i] + h * ((56 as f32 / 25 as f32) * k1_c8h18 + (283 as f32 / 14 as f32) * k2_c8h18 - (119 as f32 / 6 as f32) * k3_c8h18 - (26 as f32 / 7 as f32) * k4_c8h18 - (13 as f32 / 15 as f32) * k5_c8h18 + (149 as f32 / 32 as f32) * k6_c8h18 - (25 as f32 / 9 as f32) * k7_c8h18 + (27 as f32 / 25 as f32) * k8_c8h18));

            let k9_ic8h18 = self.k_1 * (c_c8h18_mod[i] + h * ((56 as f32 / 25 as f32) * k1_c8h18 + (283 as f32 / 14 as f32) * k2_c8h18 - (119 as f32 / 6 as f32) * k3_c8h18 - (26 as f32 / 7 as f32) * k4_c8h18 - (13 as f32 / 15 as f32) * k5_c8h18 + (149 as f32 / 32 as f32) * k6_c8h18 - (25 as f32 / 9 as f32) * k7_c8h18 + (27 as f32 / 25 as f32) * k8_c8h18)) - self.k_2 * (c_ic8h18_mod[i] + h * ((56 as f32 / 25 as f32) * k1_ic8h18 + (283 as f32 / 14 as f32) * k2_ic8h18 - (119 as f32 / 6 as f32) * k3_ic8h18 - (26 as f32 / 7 as f32) * k4_ic8h18 - (13 as f32 / 15 as f32) * k5_ic8h18 + (149 as f32 / 32 as f32) * k6_ic8h18 - (25 as f32 / 9 as f32) * k7_ic8h18 + (27 as f32 / 25 as f32) * k8_ic8h18));

            let k9_c4h10 = self.k_2 * (c_ic8h18_mod[i] + h * ((56 as f32 / 25 as f32) * k1_ic8h18 + (283 as f32 / 14 as f32) * k2_ic8h18 - (119 as f32 / 6 as f32) * k3_ic8h18 - (26 as f32 / 7 as f32) * k4_ic8h18 - (13 as f32 / 15 as f32) * k5_ic8h18 + (149 as f32 / 32 as f32) * k6_ic8h18 - (25 as f32 / 9 as f32) * k7_ic8h18 + (27 as f32 / 25 as f32) * k8_ic8h18));

            let k9_c4h8 = self.k_2 * (c_ic8h18_mod[i] + h * ((56 as f32 / 25 as f32) * k1_ic8h18 + (283 as f32 / 14 as f32) * k2_ic8h18 - (119 as f32 / 6 as f32) * k3_ic8h18 - (26 as f32 / 7 as f32) * k4_ic8h18 - (13 as f32 / 15 as f32) * k5_ic8h18 + (149 as f32 / 32 as f32) * k6_ic8h18 - (25 as f32 / 9 as f32) * k7_ic8h18 + (27 as f32 / 25 as f32) * k8_ic8h18));
```

```
            c_c8h18_mod.push(c_c8h18_mod[i] + h/840.0 * (41.0*k1_c8h18 +
216.0*k4_c8h18 + 27.0*k5_c8h18 + 272.0*k6_c8h18 + 27.0*k7_c8h18 + 216.0*k8_c8h18 +
41.0*k9_c8h18));
            c_ic8h18_mod.push(c_ic8h18_mod[i] + h/840.0 * (41.0*k1_ic8h18 +
216.0*k4_ic8h18 + 27.0*k5_ic8h18 + 272.0*k6_ic8h18 + 27.0*k7_ic8h18 + 216.0*k8_ic8h18
+ 41.0*k9_ic8h18));
            c_c4h10_mod.push(c_c4h10_mod[i] + h/840.0 * (41.0*k1_c4h10 +
216.0*k4_c4h10 + 27.0*k5_c4h10 + 272.0*k6_c4h10 + 27.0*k7_c4h10 + 216.0*k8_c4h10 +
41.0*k9_c4h10));
            c_c4h8_mod.push(c_c4h8_mod[i] + h/840.0 * (41.0*k1_c4h8 + 216.0*k4_c4h8 +
27.0*k5_c4h8 + 272.0*k6_c4h8 + 27.0*k7_c4h8 + 216.0*k8_c4h8 + 41.0*k9_c4h8));
        }
        return (c_c8h18_mod, c_ic8h18_mod, c_c4h10_mod, c_c4h8_mod);
    }
}

fn main() {
    let init = InitialData {
        temperature_1: 458.0,
        time: vec![0, 30, 60, 120, 240, 300],
        pressure: vec![4.21e3, 4.51e3, 4.73e3, 5.04e3, 5.40e3, 5.52e3],
        time_1: 250,
        universal_gas_equation_system: 8.31,
        temperature_2: 620.0,
        k_1: 0.12,
        k_2: 0.80,
        c_c8h18: 0.036e3,
        c_ic8h18: 0.0,
```

```
        c_c4h10: 0.0,
        c_c4h8: 0.0,
        dt: 0.01,
        number_of_iterations: (0..1000).collect(),
    };

    println!("Task_1:");

    let pressure_a = init.pressure_calc();
    let concentration_a = init.concentration_calc(pressure_a.clone());
    let concentration_anamorphosis_0 = init.anamorphosis0(concentration_a.clone());
    let concentration_anamorphosis_1 = init.anamorphosis1(concentration_a.clone());
    let concentration_anamorphosis_2 = init.anamorphosis2(concentration_a.clone());
    let rate_constant0 = init.rate_constant0_calc(concentration_a.clone());
    let rate_constant1 = init.rate_constant1_calc(concentration_a.clone());
    let rate_constant2 = init.rate_constant2_calc(concentration_a.clone());

    println!("\tTask_1_1:");
    println!("\t\tt, sec\t\tp_A, Pa\t\tC_A, mol/m^3");
    for i in 0..pressure_a.len() {
        println!("\t\tt[{}] = {}\tp_A[{}] = {}\tC_A[{}] = {}", i, init.time[i],
i, pressure_a[i], i, concentration_a[i]);
    }
    println!("\n\tTask_1_2:");
    println!("\t\tk0, mol/(m^3 * sec)\tk1, 1/sec\t\tk2, 1/(mol * m^3 * sec))");
    for i in 0..pressure_a.len() {
        println!("\t\tk0[{}] = {}\tk1[{}] = {}\tk2[{}] = {}", i,
rate_constant0[i], i, rate_constant1[i], i, rate_constant2[i]);
```

```
    }

    println!("\n\tTask_1_3:");

    println!("\t\ta_0, mol/m^3\t\ta_1, mol/m^3\t\ta_2, mol/m^3");

    for i in 0..pressure_a.len() {

        println!("\t\ta_0[{}] = {}\t\ta_1[{}] = {}\ta_2[{}] = {}", i,
concentration_anamorphosis_0[i], i, concentration_anamorphosis_1[i], i,
concentration_anamorphosis_2[i]);

    }


    write_to_file(concentration_a.clone(), init.time.clone(),
"D:/lab_PhysChem/Lab6/C(dt)/C(dt).txt");

    write_to_file(concentration_anamorphosis_0.clone(), init.time.clone(),
"D:/lab_PhysChem/Lab6/C_a0(dt)/C_a0(dt).txt");

    write_to_file(concentration_anamorphosis_1.clone(), init.time.clone(),
"D:/lab_PhysChem/Lab6/C_a1(dt)/C_a1(dt).txt");

    write_to_file(concentration_anamorphosis_2.clone(), init.time.clone(),
"D:/lab_PhysChem/Lab6/C_a2(dt)/C_a2(dt).txt");


    println!("\n\tTask_1_4:");

    let (half_turn_time, concentration_time_moment, tranformation_degree) =
init.time_moment(init.time[1] as f32, rate_constant2[1], concentration_a[0],
concentration_a[1]);

    println!("\n\tt(1/2) = {} sec\n\tC(t[1]) = {} mol/m^3\n\ta(t[1]) = {}",
half_turn_time, concentration_time_moment, tranformation_degree);


    let mut c_c8h18_mod: Vec<f32> = vec![];

    c_c8h18_mod.push(init.c_c8h18);

    let mut c_ic8h18_mod: Vec<f32> = vec![];

    c_ic8h18_mod.push(init.c_ic8h18);

    let mut c_c4h10_mod: Vec<f32> = vec![];
```

```
    c_c4h10_mod.push(init.c_c4h10);

    let mut c_c4h8_mod: Vec<f32> = vec![];

    c_c4h8_mod.push(init.c_c4h8);

    (c_c8h18_mod, c_ic8h18_mod, c_c4h10_mod, c_c4h8_mod) =
init.equation_system(c_c8h18_mod, c_ic8h18_mod, c_c4h10_mod, c_c4h8_mod);

    write_to_file_conc(c_c8h18_mod, init.number_of_iterations.clone(),
"D:/lab_PhysChem/Lab6/Concentrations/C_C8H18/c_c8h18(t).txt");

    write_to_file_conc(c_ic8h18_mod, init.number_of_iterations.clone(),
"D:/lab_PhysChem/Lab6/Concentrations/C_i-C8H18/c_ic8h18(t).txt");

    write_to_file_conc(c_c4h10_mod, init.number_of_iterations.clone(),
"D:/lab_PhysChem/Lab6/Concentrations/C_C4H10/c_c4h10(t).txt");

    write_to_file_conc(c_c4h8_mod, init.number_of_iterations.clone(),
"D:/lab_PhysChem/Lab6/Concentrations/C_C4H8/c_c4h8(t).txt");

}


// вывод в файл анаморфоз

fn write_to_file(vector: Vec<f32>, time: Vec<i32>, path: &str) -> std::io::Result<()>
{

    let mut file = File::create(path)?;

    for i in 0..time.len() {

        write!(file, "{} {}\n", time[i], vector[i])?;

    }

    Ok(())

}


// вывод в файл системы уравнений

fn write_to_file_conc(vector: Vec<f32>, number_of_iterations: Vec<usize>, path: &str)
-> std::io::Result<()> {

    let mut file = File::create(path)?;
```

```
    for i in 0..number_of_iterations.len() {
        write!(file, "{} {}\n", number_of_iterations[i], vector[i])?;
    }
    Ok(())
}
```

Результаты расчетов

Task_1:

Task_1_1:

t, sec p_A, Pa C_A, mol/m^3

t[0] = 0 p_A[0] = 4210 C_A[0] = 1.106154

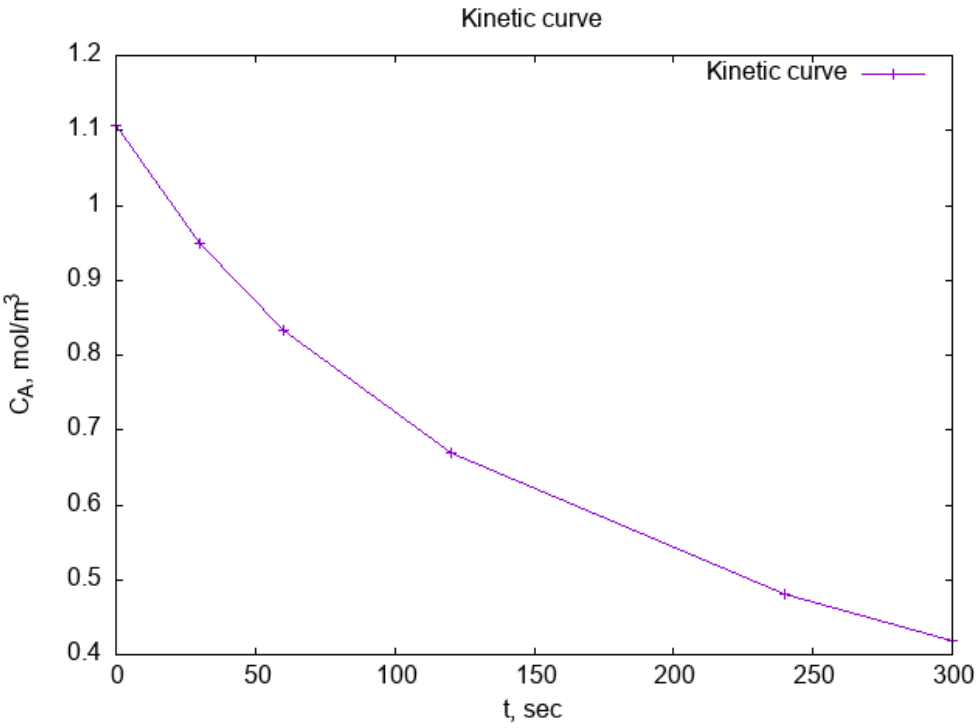
t[1] = 30 p_A[1] = 3610 C_A[1] = 0.9485073

t[2] = 60 p_A[2] = 3170 C_A[2] = 0.83289975

t[3] = 120 p_A[3] = 2550 C_A[3] = 0.6699982

t[4] = 240 p_A[4] = 1830 C_A[4] = 0.48082227

t[5] = 300 p_A[5] = 1590 C_A[5] = 0.4177636



Task_1_2:

k0, mol/(m^3 * sec) k1, 1/sec k2, 1/(mol * m^3 * sec))

k0[0] = NaN k1[0] = NaN k2[0] = NaN

k0[1] = 0.005254889 k1[1] = 0.005125165 k2[1] = 0.005008495

k0[2] = 0.0045542372 k1[2] = 0.004728852 k2[2] = 0.00494319

k0[3] = 0.0036346314 k1[3] = 0.0041780784 k2[3] = 0.004904234

k0[4] = 0.0026055488 k1[4] = 0.0034714448 k2[4] = 0.0048989053

k0[5] = 0.0022946347 k1[5] = 0.0032457623 k2[5] = 0.00496555

Из результатов аналитического метода видно, что наименьшее отклонение результатов дает третий столбец, соответственно это и есть наша константа скорости, а порядок реакции - 2.

Task_1_3:

a_0, mol/m^3 a_1, a_2, m^3/mol

a_0[0] = 0 a_1[0] = 0 a_2[0] = 0

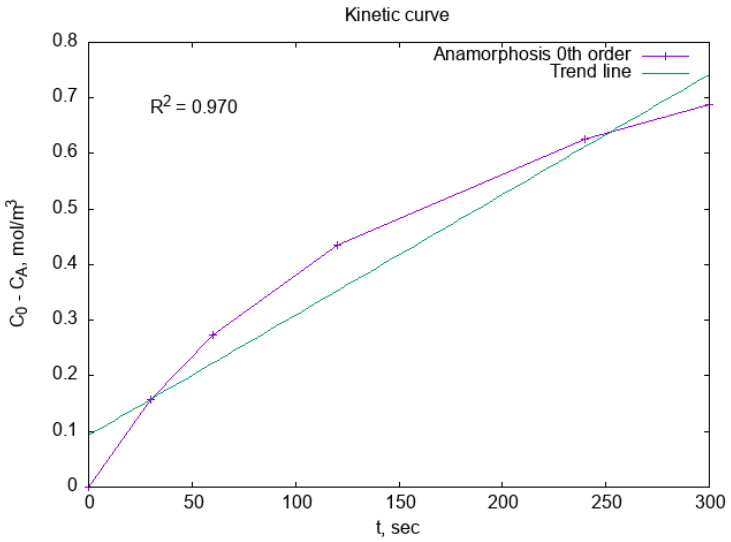
a_0[1] = 0.15764666 a_1[1] = 0.15375493 a_2[1] = 0.15025485

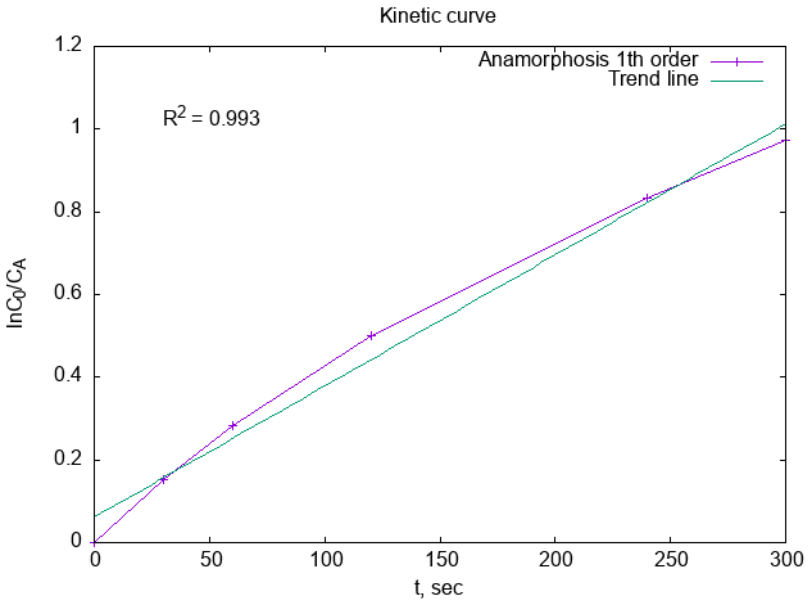
a_0[2] = 0.27325422 a_1[2] = 0.2837311 a_2[2] = 0.2965914

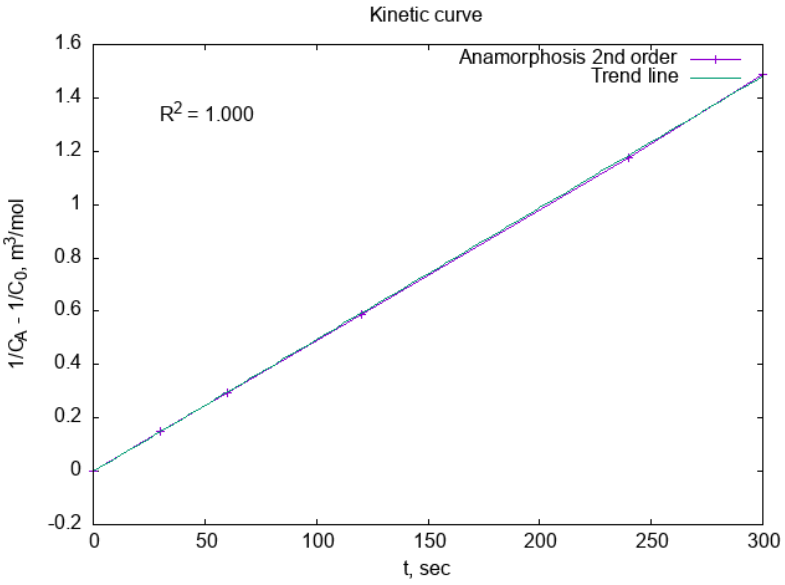
a_0[3] = 0.43615574 a_1[3] = 0.50136936 a_2[3] = 0.588508

a_0[4] = 0.6253317 a_1[4] = 0.8331467 a_2[4] = 1.1757373

a_0[5] = 0.6883904 a_1[5] = 0.97372866 a_2[5] = 1.4896649







Решая графическом способом, можно заметить, что сумма квадратов отклонений от линейной функции наблюдается на третьем графике, который является 2 порядком протекания реакции. Константа скорости реакции равна угловому коэффициенту прямой, т.е. $k = 0.00494\ c^{-1}$.

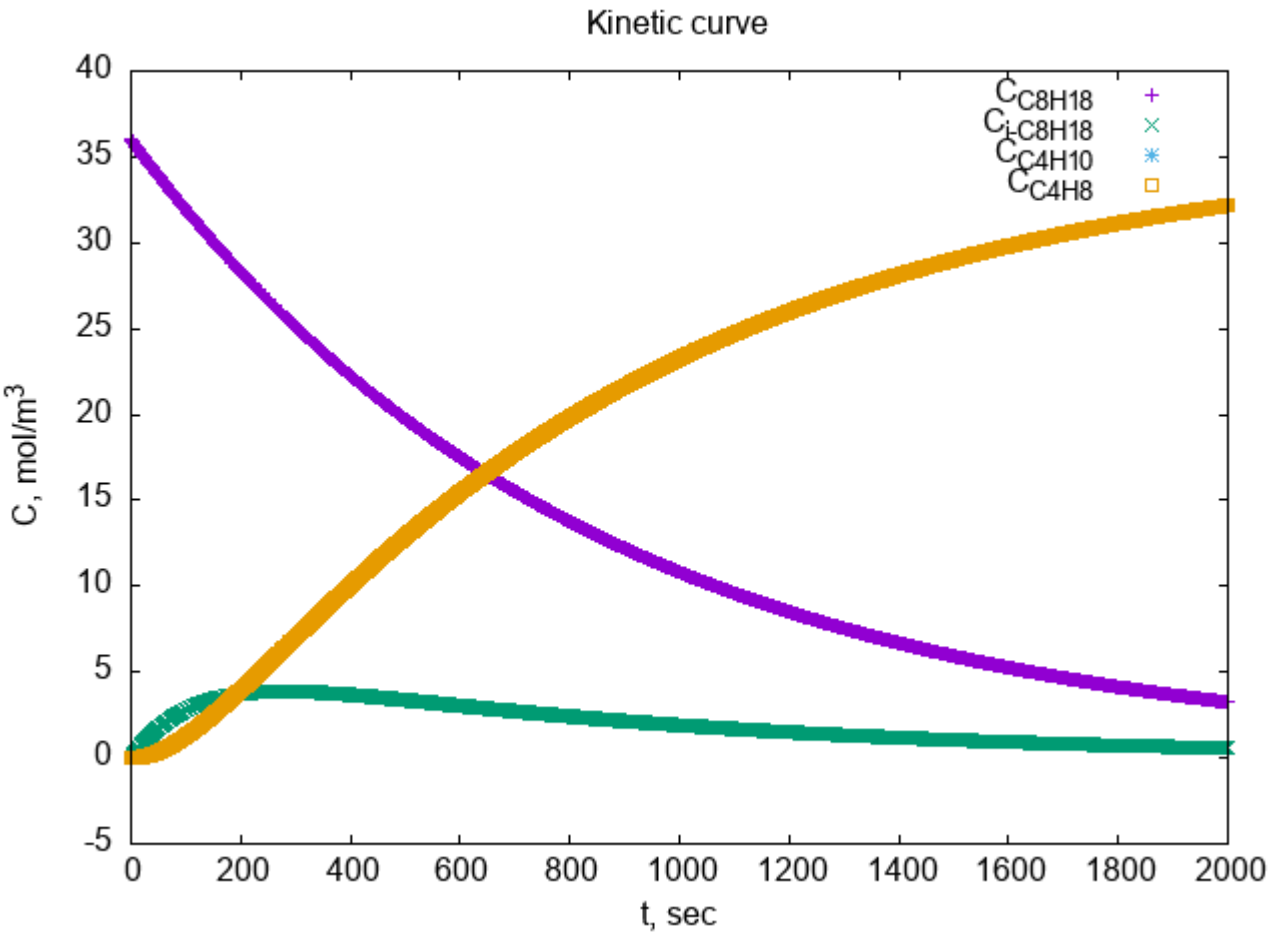
Task_1_4:

t(1/2) = 182.88458 sec

C(t_1) = 0.46732667 mol/m^3

a(t_1) = 0.57752115

Task_2:



Для продукта реакции, из-за образования промежуточного вещества, на кинетической кривой имеется точка перегиба, а скорость образования в начальный момент равна нулю, затем проходит через максимум и к концу реакции снова стремится к нулю. Кинетическая кривая промежуточного вещества имеет вид кривой с максимумом, так как в одних стадиях промежуточное вещество получается, а в других расходуется. К концу реакции концентрация промежуточного вещества уменьшается до нуля. Скорость получения промежуточного вещества с течением времени уменьшается, а скорость его расходования увеличивается, поэтому кинетическая кривая проходит через максимум. В максимуме кинетической кривой

концентрация промежуточного вещества постоянна и скорость образования равна нулю.