

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное
учреждение высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**




Инженерная школа природных ресурсов
Направление подготовки 18.04.01 «Химическая технология»
Образовательная программа «Химическая технология подготовки нефти и газа»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

По дисциплине
РУТНОН ДЛЯ ЗАДАЧ ХИМИЧЕСКОЙ ТЕХНОЛОГИИ

Студент

Группа	ФИО	Подпись	Дата
2ДМ22	Лукьянов Д.М.		21.12.2023

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОХИ ИШПР	Чузлов В.А.	к.т.н.		22.12.2023

ЗАДАНИЕ 1

Найдите минимум следующих функций, используя методы минимизации, доступные в функции `scipy.optimize.minimize()`. Начальное приближение: $x_0 = [0, 0]$.

1. Функция Экли:

$$f(x, y) = -20 \exp \left[-0.2 \sqrt{0.5(x^2 + y^2)} \right] - \exp[0.5(\cos(2\pi x) + \cos(2\pi y))] + e + 20$$

2. Функция Била:

$$f(x, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$$

3. Функция Гольдшейна-Прайса:

$$f(x, y) = [1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)] \cdot [30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)]$$

4. Функция Матьяса:

$$f(x, y) = 0.26(x^2 + y^2) - 0.48xy$$

Программная реализация (в Google Colab):

Cell 1

```
!pip install pygad
```

Cell 2

```
from scipy.optimize import minimize
from math import *
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from scipy.integrate import solve_ivp
import pygad as ga
```

Cell 3

```
def ecly(X):
    x, y = X
    x, y = np.array(x), np.array(y)
    res = -20 * np.exp(-0.2 * (0.5 * (x**2 + y**2))**0.5) - \
        np.exp(0.5 * (np.cos(2 * pi * x) + np.cos(2 * pi * y))) + np.e + 20
    return res

def Beel(X):
```

```

x, y = X
x, y = np.array(x), np.array(y)
res = (1.5 - x + x * y)**2 + (2.25 - x + x * y**2)**2 + (2.625 - \
    x + x*y**3)**2
return res

```

```

def Gold_Price(X):
    x, y = X
    x, y = np.array(x), np.array(y)
    res = (1 + (x + y + 1)**2 * (19 - 14 * x + 3 * x**2 - 14 * y + 6 * x * y + \
        3 * y**2)) * (30 + (2 * x - 3 * y)**2 * (18 - 32 * x + 12 * x**2 + \
        48 * y - 36 * x * y + 27 * y**2))
    return res

```

```

def Mat(X):
    x, y = X
    x, y = np.array(x), np.array(y)
    res = 0.26 * (x**2 + y**2) - 0.48 * x * y
    return res

```

Cell 4

```

answ = minimize(ecly, (0, 0), method='BFGS')
answ

```

Cell 5

```

answ = minimize(ecly, (0, 0), method='Nelder-Mead')
answ

```

Cell 6

```

answ = minimize(ecly, (0, 0), method='CG')
answ

```

Cell 7

```

span = 1
x = np.linspace(-span, span, 100)
y = np.linspace(-span, span, 100)
cmap = matplotlib.cm.magma_r
fig = plt.figure(figsize=(8,6), dpi=450)
ax = fig.add_subplot(xlim=[x[0], x[-1]], ylim=[y[0], y[-1]])
X, Y = np.meshgrid(x, y)
Z = ecly((X, Y))
ax.contourf(X, Y, Z, cmap=cmap)
A = matplotlib.colors.Normalize(np.min(Z), np.max(Z))
cbar = fig.colorbar(matplotlib.cm.ScalarMappable(norm=A, cmap=cmap), ax=ax);

```

Cell 8
<pre>answ = minimize(Beel, (0, 0), method='BFGS') answ</pre>
Cell 9
<pre>answ = minimize(Beel, (0, 0), method='Nelder-Mead') answ</pre>
Cell 10
<pre>answ = minimize(Beel, (0, 0), method='CG') answ</pre>
Cell 11
<pre>x = np.linspace(2.95, 3.05, 100) y = np.linspace(0.45, 0.55, 100) cmap = matplotlib.cm.magma_r fig = plt.figure(figsize=(8,6), dpi=450) ax = fig.add_subplot(xlim=[x[0], x[-1]], ylim=[y[0], y[-1]]) X, Y = np.meshgrid(x, y) Z = Beel((X, Y)) ax.contourf(X, Y, Z, cmap=cmap) A = matplotlib.colors.Normalize(np.min(Z), np.max(Z)) cbar = fig.colorbar(matplotlib.cm.ScalarMappable(norm=A, cmap=cmap), ax=ax);</pre>
Cell 12
<pre>answ = minimize(Gold_Price, (0, 0), method='BFGS') answ</pre>
Cell 13
<pre>answ = minimize(Gold_Price, (0, 0), method='Nelder-Mead') answ</pre>
Cell 14
<pre>answ = minimize(Gold_Price, (0, 0), method='CG') answ</pre>
Cell 15
<pre>span = 5 x = np.linspace(-0.7, -0.5, 100) y = np.linspace(-0.5, -0.3, 100) cmap = matplotlib.cm.magma_r fig = plt.figure(figsize=(8,6), dpi=450) ax = fig.add_subplot(xlim=[x[0], x[-1]], ylim=[y[0], y[-1]]) X, Y = np.meshgrid(x, y) Z = Gold_Price((X, Y)) ax.contourf(X, Y, Z, cmap=cmap)</pre>

```
A = matplotlib.colors.Normalize(np.min(Z), np.max(Z))
cbar = fig.colorbar(matplotlib.cm.ScalarMappable(norm=A, cmap=cmap), ax=ax);
```

Cell 16

```
answ = minimize(Mat, (0, 0), method='BFGS')
answ
```

Cell 17

```
answ = minimize(Mat, (0, 0), method='Nelder-Mead')
answ
```

Cell 18

```
answ = minimize(Mat, (0, 0), method='CG')
answ
```

Cell 19

```
span = 1
x = np.linspace(-span, span, 100)
y = np.linspace(-span, span, 100)
cmap = matplotlib.cm.magma_r
fig = plt.figure(figsize=(8,6), dpi=450)
ax = fig.add_subplot(xlim=[x[0], x[-1]], ylim=[y[0], y[-1]])
X, Y = np.meshgrid(x, y)
Z = Mat((X, Y))
ax.contourf(X, Y, Z, cmap=cmap)
A = matplotlib.colors.Normalize(np.min(Z), np.max(Z))
cbar = fig.colorbar(matplotlib.cm.ScalarMappable(norm=A, cmap=cmap), ax=ax);
```

Ответ:

Минимум функции Экли:

1. По методу BFGS:

$$f_{min} = f(0,0) = 0$$

2. По методу Нельдера-Мида

$$f_{min} = f(0,0) = 0$$

3. По методу CG:

$$f_{min} = f(0,0) = 0$$

Глобальный минимум для функции Экли найден

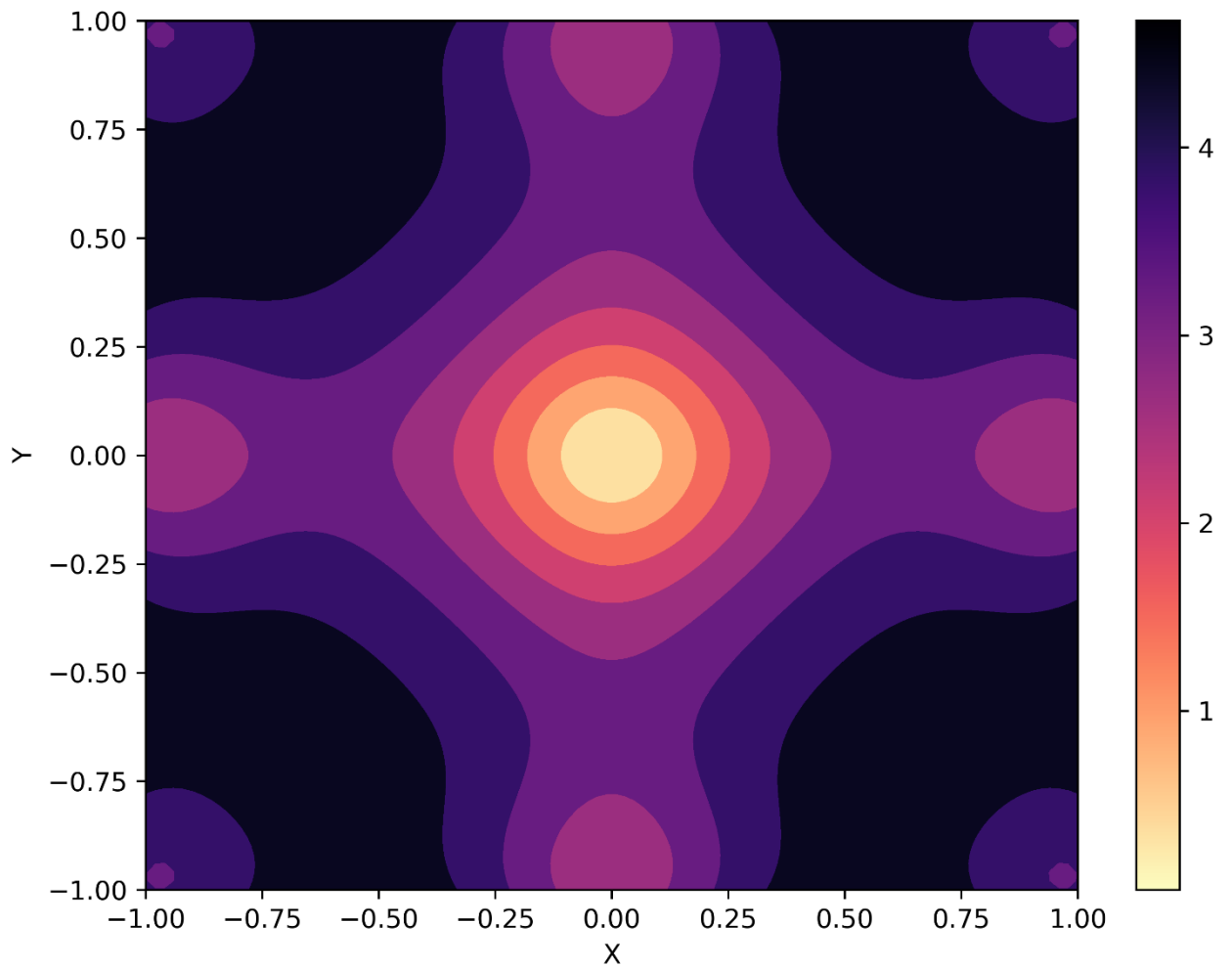


Рисунок 1 – Функция Экли

Минимум функции Била:

1. По методу BFGS

$$f_{min} = f(3.0, 0.5) = 0$$

2. По методу Нельдера-Мида

$$f_{min} = f(3.0, 0.5) = 0$$

3. По методу CG:

$$f_{min} = f(3.0, 0.5) = 0$$

Глобальный минимум для функции Била найден

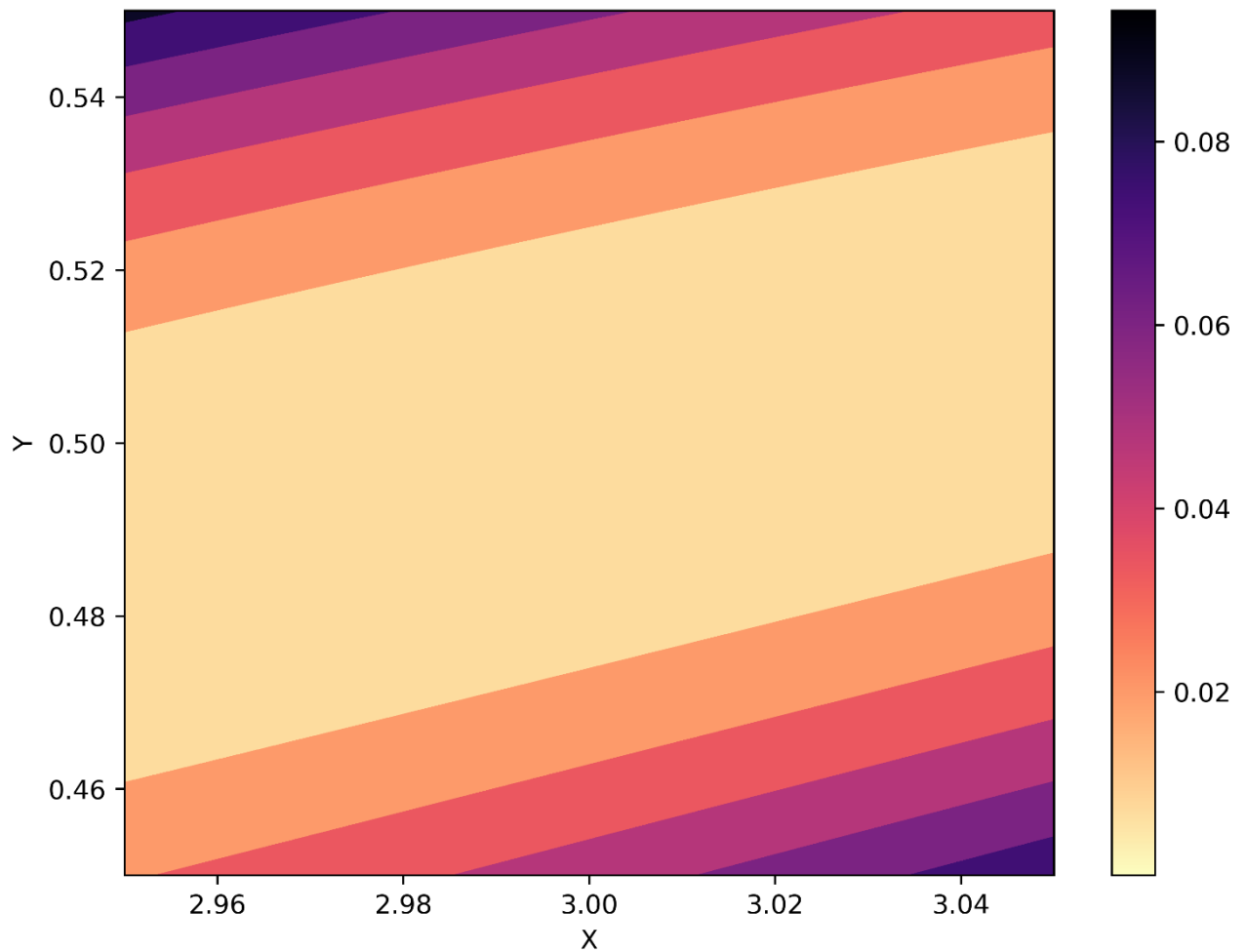


Рисунок 2 – Функция Била

Минимум функции Гольдштейна-Прайса:

4. По методу BFGS

$$f_{min} = f(-0,6, -0,4) = 30$$

5. По методу Нельдера-Мида

$$f_{min} = f(-0,6, -0,4) = 30$$

6. По методу CG:

$$f_{min} = f(-0,6, -0,4) = 30$$

Глобальный минимум для функции Гольдштейна-Прайса не найден при начальном приближении $x_0 = [0, 0]$

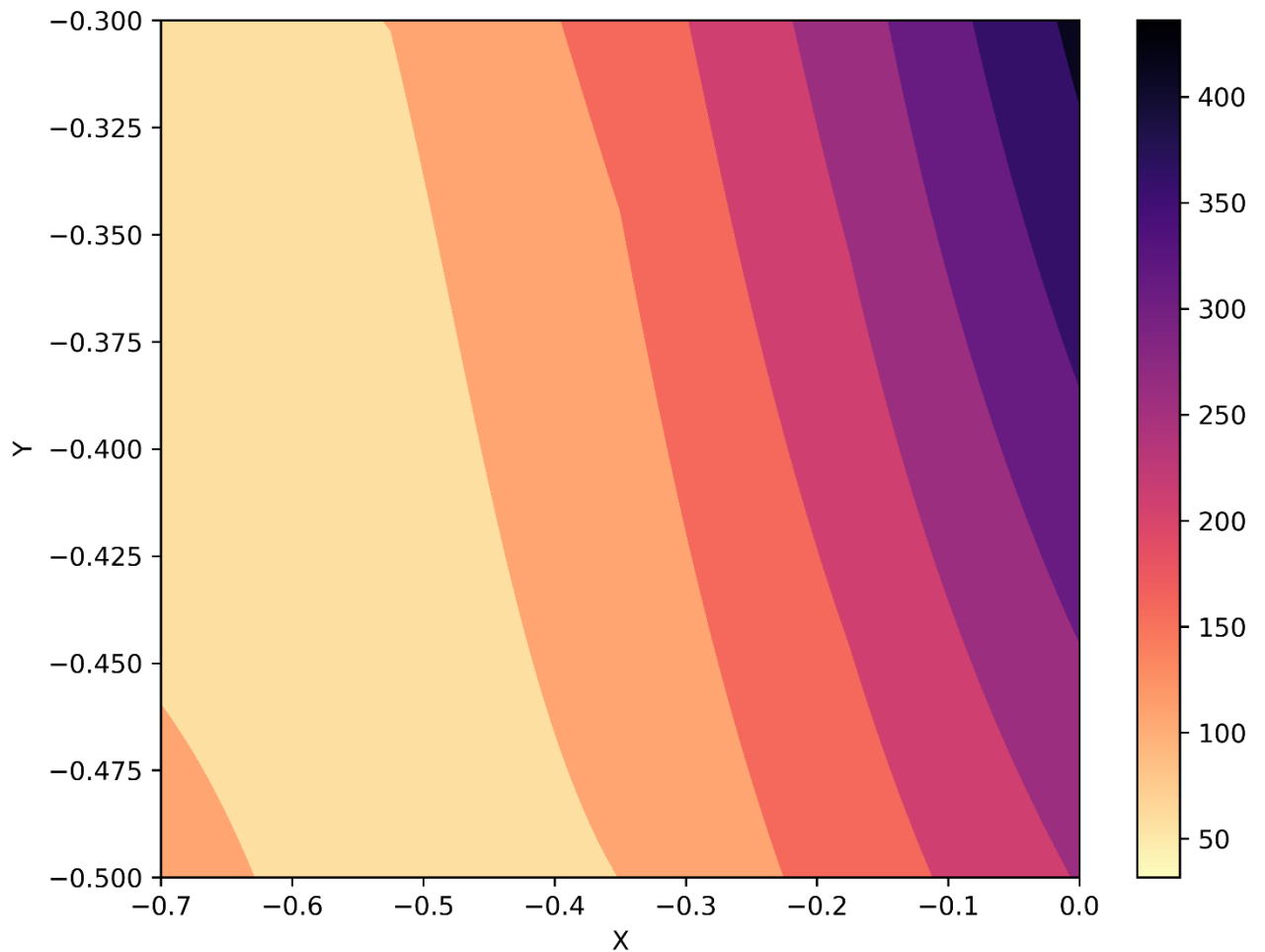


Рисунок 3 – Функция Гольдшейна-Прайса

Минимум функции Матьяса:

7. По методу BFGS

$$f_{min} = f(0,0) = 0$$

8. По методу Нельдера-Мида

$$f_{min} = f(0,0) = 0$$

9. По методу CG:

$$f_{min} = f(0,0) = 0$$

Глобальный минимум для функции Матьяса найден

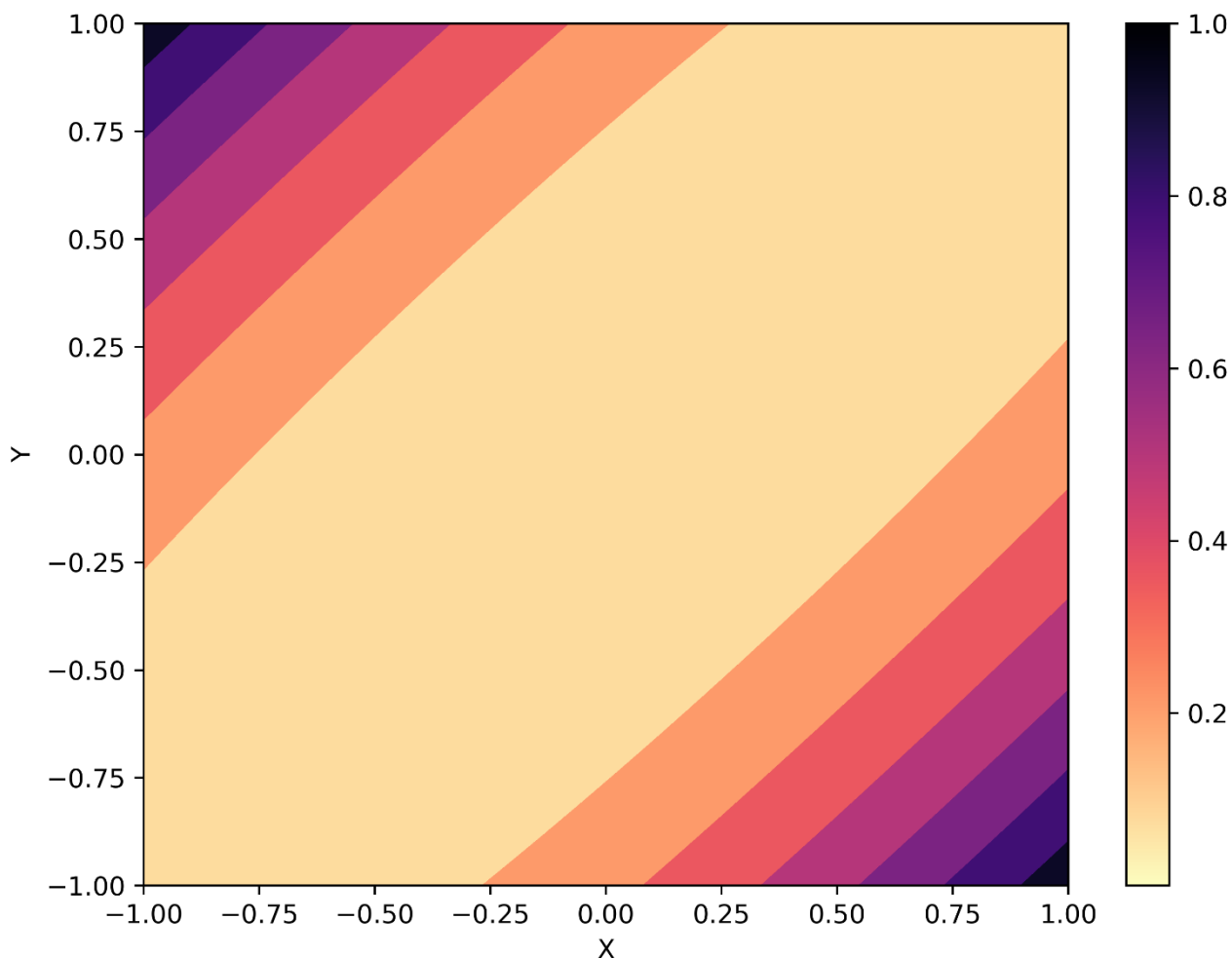
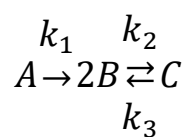


Рисунок 4 – Функция Матьясаф

ЗАДАНИЕ 2

Пусть дана схема химических превращений:



Необходимо определить с помощью генетического алгоритма и метода Нелдера-Мида (можно воспользоваться функцией `scipy.optimize.minimize()`, указав соответствующее значение опционального аргумента `method`) константы скоростей реакции: k_1, k_2 и k_3 , если известно, что к моменту времени $t = 1(\text{с})$ концентрации компонентов равны: $C_A = 0.1423, C_B = 1.5243, C_C = 0.5956$ моль/л.

Начальные условия: $C_A(0) = 1.0; C_B = 0.0; C_C = 0.5$ моль/л

Область поиска для всех констант ограничте интервалом [0; 2].

Програмная реализация (в Google Colab):

Cell 20

```
t_start, t_end, h = 0, 1, 0.01
t = np.arange(t_start, t_end+h, h)
initial_composition = [1.0, 0.0, 0.5]
actual_values = [0.1423, 1.5243, 0.5956]
k = (1.0, 1.0, 1.0)

def equations(t, c, k):
    right_parts = [
        (-1) * k[0] * c[0],
        2 * k[0] * c[0] - 2 * k[1] * c[1]**2 + 2 * k[2] * c[2],
        k[1] * c[1]**2 - k[2] * c[2]
    ]
    return right_parts

def obj_func(ga_instance, k, sol_ind=0):
    # решение системы ОДУ численным методом

    sol = solve_ivp(equations, (t_start, t_end), initial_composition, args=(k,))
    c = sol.y
    return 1/sum((c[i][-1] - actual_values[i]) ** 2
                 for i in range(len(actual_values)))

genes_spaces = [{'low': 0.001, 'high': 2},
                 {'low': 0.001, 'high': 2},
                 {'low': 0.001, 'high': 2},]

ga_instance = ga.GA(50, 20, obj_func, sol_per_pop=100, num_genes=3,
                    mutation_num_genes=1, init_range_low=0, gene_space=genes_spaces)
ga_instance.run()
solution, solution_fitness, solution_idx = ga_instance.best_solution()
print(f"Parameters of the best solution : {solution}")
print(f"Fitness value of the best solution = {solution_fitness}")
```

Cell 21

```
t_start, t_end, h = 0, 2, 0.01
t = np.arange(t_start, t_end+h, h)
sol = solve_ivp(equations, (t_start, t_end), initial_composition, t_eval=t,
```

```

        args=(solution, ))
c_a, c_b, c_c, t_x = sol.y[0], sol.y[1], sol.y[2], sol.t

t_exp = [1, 1, 1]
xlim = [t[0], t[-1]]
fig = plt.figure(figsize=(8,6), dpi=450)
ax = fig.add_subplot(xlim=xlim)

ax.plot(t_x, c_a, 'b', label='$C_a$')
ax.plot(t_x, c_b, 'r', label='$C_b$')
ax.plot(t_x, c_c, 'g', label='$C_c$')
ax.scatter(t_exp[0], actual_values[0], c='b', label='$C_a$ эксп')
ax.scatter(t_exp[1], actual_values[1], c='r', label='$C_b$ эксп')
ax.scatter(t_exp[2], actual_values[2], c='g', label='$C_c$ эксп')

ax.legend()
ax.set_ylabel('Концентрация, моль/л')
ax.set_xlabel('Время, с');

```

Cell 22

```

t_start, t_end, h = 0, 1, 0.01
t = np.arange(t_start, t_end+h, h)
k1bounds = (0, 2)
k2bounds = (0, 2)
k3bounds = (0, 2)
bounds = k1bounds, k2bounds, k3bounds

def obj_func2(k):
    # решение системы ОДУ численным методом

    sol = solve_ivp(equations, (t_start, t_end), initial_composition, args=(k,))
    c = sol.y
    return sum((c[i][-1] - actual_values[i]) ** 2
               for i in range(len(actual_values)))

solution2 = minimize(obj_func2, (1, 1, 1), method='Nelder-Mead', bounds=bounds)
print(solution2)

```

Cell 23

```

t_start, t_end, h = 0, 2, 0.01
t = np.arange(t_start, t_end+h, h)
sol = solve_ivp(equations, (t_start, t_end), initial_composition, t_eval=t,
                args=(solution2.x, ))

```

```

c_a, c_b, c_c, t_x = sol.y[0], sol.y[1], sol.y[2], sol.t

t_exp = [1, 1, 1]
xlim = [t[0], t[-1]]
fig = plt.figure(figsize=(8,6), dpi=450)
ax = fig.add_subplot(xlim=xlim)

ax.plot(t_x, c_a, 'b', label='$C_a$')
ax.plot(t_x, c_b, 'r', label='$C_b$')
ax.plot(t_x, c_c, 'g', label='$C_c$')
ax.scatter(t_exp[0], actual_values[0], c='b', label='$C_a$ эксп')
ax.scatter(t_exp[1], actual_values[1], c='r', label='$C_b$ эксп')
ax.scatter(t_exp[2], actual_values[2], c='g', label='$C_c$ эксп')

ax.legend()
ax.set_ylabel('Концентрация, моль/л')
ax.set_xlabel('Время, с');

```

Cell 24

```

print(f"Константы по методу ГА = {solution}")
print(f"Константы по методу Нелдера-Мида = {solution2.x}")

```

Ответ:

Константы по методу ГА = [1.95627708 0.54764706 1.69736111]

Константы по методу Нелдера-Мида = [1.95009038 0.33121762 0.82966647]

Можно видеть, что подобранные значения не совпадают, что связано с недостаточным количеством экспериментальных значений. В действительности, оба метода решили задачу оптимизации корректно с математической точки зрения (рисунок 5-6).

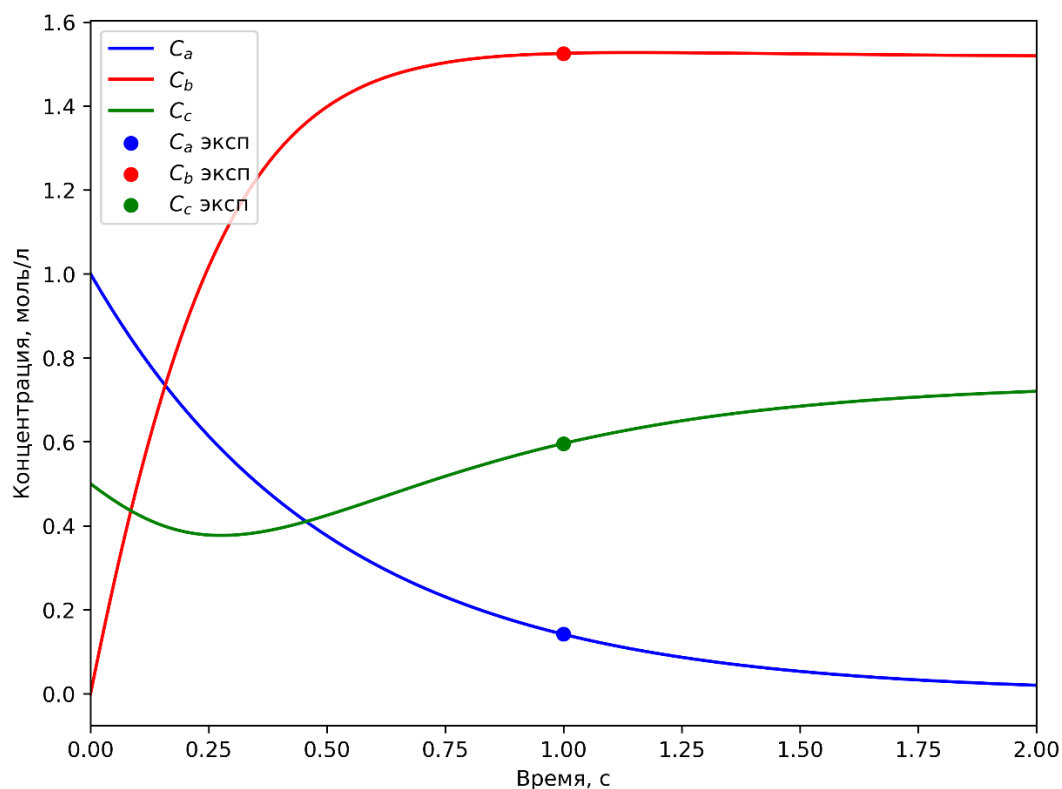


Рисунок 5 – Профиль концентраций, полученный при использовании констант скоростей, подобранных по методу ГА

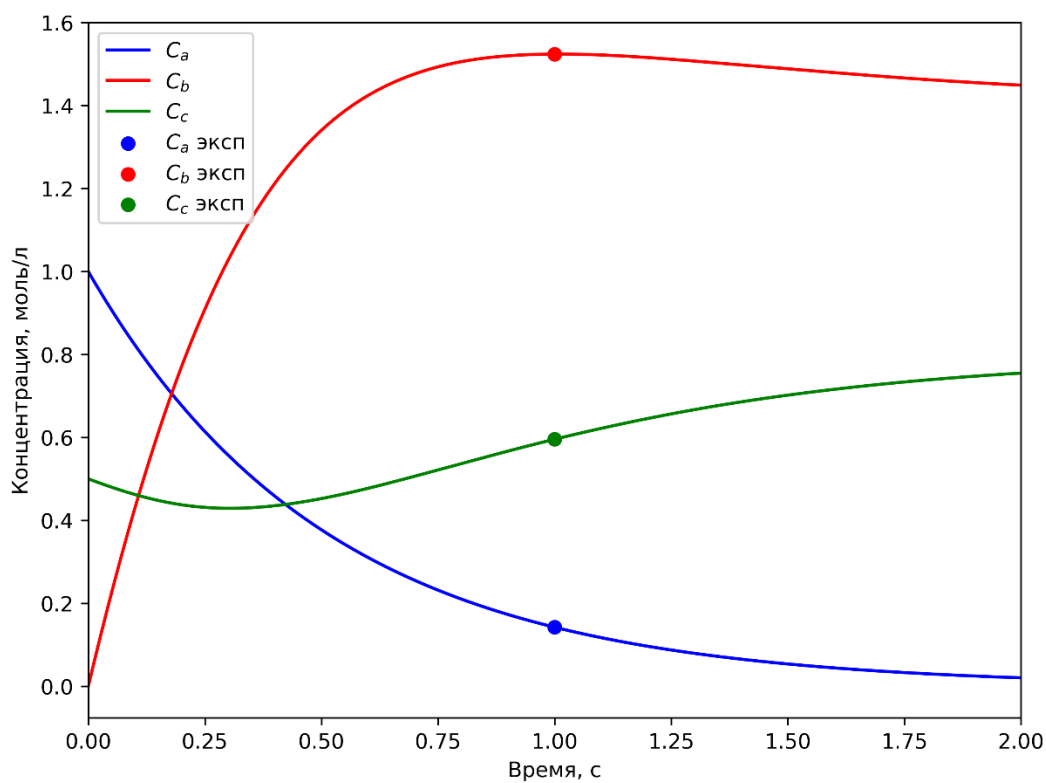


Рисунок 6 – Профиль концентраций, полученный при использовании констант скоростей, подобранных по методу Нельдера-Мида