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 № 4

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

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

1. Функция Экли
2. Функция Била
3. Функция Гольдшейна-Прайса
4. Функция Матьяса

Программная реализация:

```
import numpy as np
from scipy.optimize import minimize

print ()
initial_guess = [0,0]
print ('Initial guess is', initial_guess)
print ()

def ackley(X):
    x, y = X
    return -20.0 * np.exp(-0.2 * np.sqrt(0.5 * (x**2 + y**2))) - np.exp(0.5 *
(np.cos(2 * np.pi * x) + np.cos(2 * np.pi * y))) + np.e + 20

print ('Ackley by Nelder-Mead')
res_a = minimize(ackley, initial_guess, method = 'Nelder-Mead')
print (res_a.x)
print ()
print ('Ackley by Powell')
res_a = minimize(ackley, initial_guess, method = 'Powell')
print (res_a.x)
print ()
print ('Ackley by slsqp')
res_a = minimize(ackley, initial_guess, method = 'slsqp')
print (res_a.x)
print ()
print ('Ackley by BFGS')
res_a = minimize(ackley, initial_guess, method = 'BFGS')
print (res_a.x)
print ()
print ('Ackley by CG')
res_a = minimize(ackley, initial_guess, method = 'CG')
print (res_a.x)
print ()
print ('Ackley by TNC')
res_a = minimize(ackley, initial_guess, method = 'TNC')
```

```
print (res_a.x)
print ()
print ('Ackley by l-bfgs-b')
res_a = minimize(ackley, initial_guess, method = 'l-bfgs-b')
print (res_a.x)
print ()

def beale(X):
    x, y = X
    return (1.5 - x + x * y) ** 2 + (2.25 - x + x * y ** 2) ** 2 + (2.625 - x + x
* y ** 3) ** 2

print ('Beale by Nelder-Mead')
res_b = minimize(beale, initial_guess, method = 'Nelder-Mead')
print (res_b.x)
print ()
print ('Beale by Powell')
res_b = minimize(beale, initial_guess, method = 'Powell')
print (res_b.x)
print ()
print ('Beale by slsqp')
res_b = minimize(beale, initial_guess, method = 'slsqp')
print (res_b.x)
print ()
print ('Beale by BFGS')
res_b = minimize(ackley, initial_guess, method = 'BFGS')
print (res_b.x)
print ()
print ('Beale by CG')
res_b = minimize(ackley, initial_guess, method = 'CG')
print (res_b.x)
print ()
print ('Beale by TNC')
res_b = minimize(ackley, initial_guess, method = 'TNC')
print (res_b.x)
print ()
print ('Beale by l-bfgs-b')
res_b = minimize(ackley, initial_guess, method = 'l-bfgs-b')
print (res_b.x)
print ()

def g_p(X):
    x, y = X
    return (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))

print ('Goldstein-Price by Nelder-Mead')
res_g_p = minimize(g_p, initial_guess, method = 'Nelder-Mead')
print (res_g_p.x)
print ()
```

```
print ('Goldstein-Price by Powell')
res_g_p = minimize(g_p, initial_guess, method = 'Powell')
print (res_g_p.x)
print ()
print ('Goldstein-Price by slsqp')
res_g_p = minimize(g_p, initial_guess, method = 'slsqp')
print (res_g_p.x)
print ()
print ('Goldstein-Price by BFGS')
res_g_p = minimize(g_p, initial_guess, method = 'BFGS')
print (res_g_p.x)
print ()
print ('Goldstein-Price by CG')
res_g_p = minimize(g_p, initial_guess, method = 'CG')
print (res_g_p.x)
print ()
print ('Goldstein-Price by TNC')
res_g_p = minimize(g_p, initial_guess, method = 'TNC')
print (res_g_p.x)
print ()
print ('Goldstein-Price by l-bfgs-b')
res_g_p = minimize(g_p, initial_guess, method = 'l-bfgs-b')
print (res_g_p.x)
print ()

def matyas(X):
    x, y = X
    return 0.26 * (x ** 2 + y ** 2) - 0.48 * x * y

print ('Matyas by Nelder-Mead')
res_m = minimize(matyas, initial_guess, method = 'Nelder-Mead')
print (res_m.x)
print ()
print ('Matyas by Powell')
res_m = minimize(matyas, initial_guess, method = 'Powell')
print (res_m.x)
print ()
print ('Matyas by slsqp')
res_m = minimize(matyas, initial_guess, method = 'slsqp')
print (res_m.x)
print ()
print ('Matyas by BFGS')
res_m = minimize(matyas, initial_guess, method = 'BFGS')
print (res_m.x)
print ()
print ('Matyas by CG')
res_m = minimize(matyas, initial_guess, method = 'CG')
print (res_m.x)
print ()
print ('Matyas by TNC')
res_m = minimize(matyas, initial_guess, method = 'TNC')
```

```
print (res_m.x)
print ()
print ('Matyas by l-bfgs-b')
res_m = minimize(matyas, initial_guess, method = 'l-bfgs-b')
print (res_m.x)
print ()
```

Ответ:

```
Initial guess is [0, 0]

Ackley by Nelder-Mead
[0. 0.]

Ackley by Powell
[0. 0.]

Ackley by slsqp
[-9.01466425e-11 -9.01469759e-11]

Ackley by BFGS
[0. 0.]

Ackley by CG
[0. 0.]

Ackley by TNC
[0. 0.]

Ackley by l-bfgs-b
[0. 0.]

Beale by Nelder-Mead
[2.99994196 0.49998485]

Beale by Powell
[3.  0.5]

Beale by slsqp
[2.99893098 0.49982984]

Beale by BFGS
[0. 0.]

Beale by CG
[0. 0.]

Beale by TNC
[0. 0.]

Beale by l-bfgs-b
```

```
[0. 0.]

Goldstein-Price by Nelder-Mead
[-0.59995602 -0.40003256]

Goldstein-Price by Powell
[-0.59999999 -0.40000001]

Goldstein-Price by slsqp
[-0.60000137 -0.39999581]

Goldstein-Price by BFGS
[-0.6 -0.4]

Goldstein-Price by CG
[-0.6 -0.4]

Goldstein-Price by TNC
[-0.60000076 -0.39999936]

Goldstein-Price by l-bfgs-b
[-0.6 -0.4]

Matyas by Nelder-Mead
[0. 0.]

Matyas by Powell
[0. 0.]

Matyas by slsqp
[0. 0.]

Matyas by BFGS
[0. 0.]

Matyas by CG
[0. 0.]

Matyas by TNC
[0. 0.]

Matyas by l-bfgs-b
[0. 0.]
```

Задание 2

Пусть дана схема химических превращений:

$$A \xrightarrow{k_1} 2B \underset{k_3}{\overset{k_2}{\leftrightarrow}} C$$

Необходимо определить с помощью генетического алгоритма и метода Нелдера-Мида (можно воспользоваться функцией `scipy.optimize.mnimize()`, указав соответствующее значение опционального аргумента `method`) константы скоростей реакций: k_1, k_2 и k_3, если известно, что к моменту времени t = 1(c) 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 ограничьте интервалом [0; 2].

Программная реализация:

```
import genetic_algorithm as ga
from solve_ode import rk
def equations(t, c, k):
    right_parts = [-k[0]*c[0],
                   k[0]*c[0]-2*k[1]*c[1]**2 + k[2]*c[2],
                   2*k[1]*c[1]**2 - k[2]*c[2]]
    return right_parts

def obj_func(k, equations, method, t, h, initial_composition, actual_values):

    c = method(equations, t[0], t[-1], initial_composition, h, args=(k,))
    return sum((c[-1][i] - actual_values[i])**2 for i in
range(len(actual_values)))
actual_values = [0.1423, 1.5243, 0.5956]

k = ga.genetic_algorithm([[0, 2], [0, 2], [0, 2]], obj_func,
                         args = (equations, rk, [0,1], 0.1, [1, 0, 0.5],
actual_values))

print(k[0])
```

```
[3.118506118259779, 0.030160926891311925, 0.7891913142037632]
```

```
import genetic_algorithm as ga
import nelder_mead as n_m
from solve_ode import rk
from scipy.optimize import minimize
from functools import partial

def equations(t, c, k):
    right_parts = [-k[0]*c[0],
```

```
                  k[0]*c[0]-2*k[1]*c[1]**2 + k[2]*c[2],
                  2*k[1]*c[1]**2 - k[2]*c[2]]
    return right_parts

def obj_func(k, equations, method, t, h, initial_composition, actual_values):

    c = method(equations, t[0], t[-1], initial_composition, h, args=(k,))
    return sum((c[-1][i] - actual_values[i])**2 for i in
range(len(actual_values)))
actual_values = [0.1423, 1.5243, 0.5956]

'''
k = n_m.nelder_mead(lambda x: obj_func(x, equations, rk, [0,1], 0.1, [1, 0, 0.5],
actual_values), [[2., 0., 0.], [2., 0., 1.], [2.04, 0.1, 1.], [2., 0., 1.5]])
print(k)
obj = partial(obj_func, equations=equations, method=rk, t=[0., 1.], h=0.1,
initial_composition=[1, 0, 0.5], actual_values=actual_values)
k = n_m.nelder_mead(obj, [[2., 0., 0.], [2., 0., 1.], [2.04, 0.1, 1.], [2., 0.,
1.5]])
print(k)
'''

minimize(obj_func, (2., 0.15, 2.), args = (equations, rk, [0,1], 0.1, [1, 0,
0.5], actual_values),
        method = 'Nelder-Mead', bounds = [[0, 2], [0, 2], [0, 2]])
```

Ответ:

```
message: Optimization terminated successfully.
       success: True
        status: 0
           fun: 0.28524183663630753
             x: [ 2.000e+00  1.821e-01  2.000e+00]
           nit: 28
          nfev: 55
 final_simplex: (array([[ 2.000e+00,  1.821e-01,  2.000e+00],
                        [ 2.000e+00,  1.820e-01,  2.000e+00],
                        [ 2.000e+00,  1.820e-01,  2.000e+00],
                        [ 2.000e+00,  1.821e-01,  2.000e+00]]), array([ 2.852e-01,
2.852e-01,  2.852e-01,  2.852e-01]))
```