Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

Кафедра теоретической и прикладной информатики

Лабораторная работа №4
по дисциплине «Методы оптимизации»

Факультет:	ПМИ
Группа:	ПМ-92
Бригада:	7
Студенты:	Иванов В., Кутузов И.
Преподаватель:	Филиппова Е.В.

Новосибирск

2022

1. Цель работы

Ознакомиться со статистическими методами поиска при решении задач нелинейного программирования. Изучить методы случайного поиска при определении глобального экстремума функции.

2. Задание

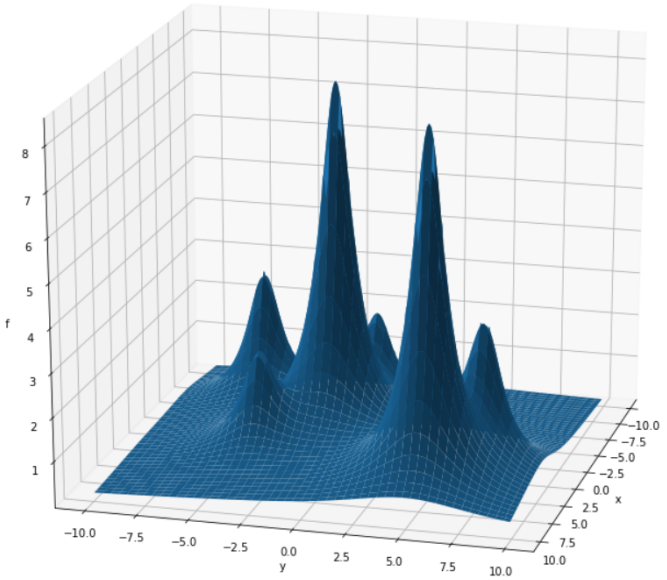
I. Разработать программу для решения задачи поиска глобального экстремума с использованием метода простого случайного поиска и трех алгоритмов глобального поиска;
II. Исследовать метод простого случайного поиска глобального экстремума при различных ε и P;
III. Исследовать алгоритмы поиска глобального экстремума. Сравнить результаты поиска по количеству вычислений функции и найденной точке экстремума. Исследование провести при различных значениях числа попыток m;
IV. Предыдущий пункт повторить при пяти разных начальных значениях ГСЧ. Сделать выводы об устойчивости различных алгоритмов.

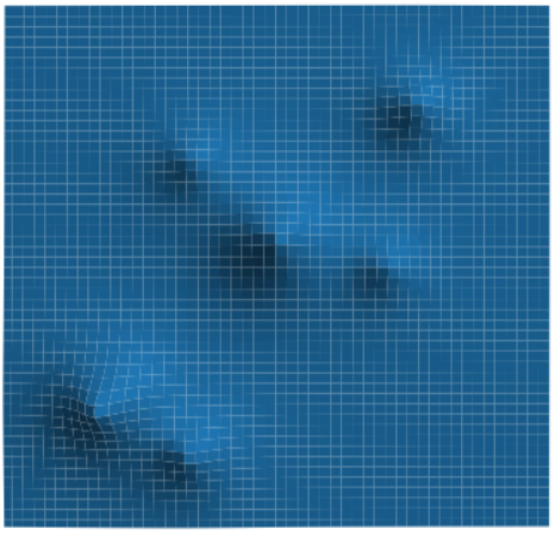
Функция для поиска максимума:

$$f(x, y) = \sum_{i=1}^{6} \frac{C_i}{1 + (x - a_i)^2 + (y - b_i)^2}$$

$$-10 \le x \le 10, \quad -10 \le y \le 10$$

C1	C2	C3	C4	C5	C6	a1	a2	a3	a4	a5	a6	b1	b2	b3	b4	b5	b6
2	3	8	3	2	8	3	-5	0	3	-4	6	-4	-6	-1	7	0	5





3. Исследования

Простой случайный поиск

	eps	P	N	x	y	max
1	0.5	0.85	7	-9.13e-02	-1.01e+00	8.359197e+00
2	0.5	0.95	11	-9.13e-02	-1.01e+00	8.359197e+00
3	0.5	0.99	17	-9.13e-02	-1.01e+00	8.359197e+00
4	0.5	0.999	25	-9.13e-02	-1.01e+00	8.359197e+00
5	0.2	0.85	47	-9.13e-02	-1.01e+00	8.359197e+00
6	0.2	0.95	74	-9.13e-02	-1.01e+00	8.359197e+00
7	0.2	0.99	113	-9.13e-02	-1.01e+00	8.359197e+00
8	0.2	0.999	170	-9.13e-02	-1.01e+00	8.359197e+00
9	0.1	0.85	189	-9.13e-02	-1.01e+00	8.359197e+00
10	0.1	0.95	299	-9.13e-02	-1.01e+00	8.359197e+00
11	0.1	0.99	459	-9.13e-02	-1.01e+00	8.359197e+00
12	0.1	0.999	688	-9.13e-02	-1.01e+00	8.359197e+00
13	**0.01**	**0.85**	**18971**	4.93e-03	-1.00e+00	**8.425099e+00**
14	0.01	0.95	29956	4.93e-03	-1.00e+00	8.425099e+00
15	0.01	0.99	46050	4.93e-03	-1.00e+00	8.425099e+00
16	0.01	0.999	69075	4.93e-03	-1.00e+00	8.425099e+00

Алгоритм 1

	m	f	x	y	max	seed
1	**2**	**4**	3.04e+00	6.97e+00	**8.425330e+00**	1
2	5	7	5.99e+00	5.00e+00	8.425330e+00	1
3	10	13	-4.42e-04	-1.00e+00	8.425330e+00	1
4	20	23	5.99e+00	5.00e+00	8.425330e+00	1
5	50	55	-4.23e-04	-1.00e+00	8.425330e+00	1

6	100	105	-4.99e+00	-5.99e+00	8.425330e+00	1
1	2	4	2.96e+00	-3.96e+00	8.374528e+00	2
2	**5**	**8**	5.99e+00	5.00e+00	**8.425330e+00**	2
3	10	14	-4.40e-04	-1.00e+00	8.425330e+00	2
4	20	24	5.99e+00	5.00e+00	8.425330e+00	2
5	50	55	-4.99e+00	-5.99e+00	8.425330e+00	2
6	100	107	-4.12e-04	-1.00e+00	8.425330e+00	2
1	2	5	-4.99e+00	-5.99e+00	3.736899e+00	3
2	**5**	**9**	3.04e+00	6.97e+00	**8.425330e+00**	3
3	10	15	5.99e+00	5.00e+00	8.425330e+00	3
4	20	25	3.04e+00	6.97e+00	8.425330e+00	3
5	50	56	-4.17e-04	-1.00e+00	8.425330e+00	3
6	100	107	-4.99e+00	-5.99e+00	8.425330e+00	3
1	2	3	2.96e+00	-3.96e+00	3.284624e+00	4
2	**5**	**8**	-4.99e+00	-5.99e+00	**8.425330e+00**	4
3	10	13	5.99e+00	5.00e+00	8.425330e+00	4
4	20	25	5.99e+00	5.00e+00	8.425330e+00	4
5	50	56	-4.17e-04	-1.00e+00	8.425330e+00	4
6	100	106	2.96e+00	-3.96e+00	8.425330e+00	4
1	**2**	**5**	5.99e+00	5.00e+00	**8.425330e+00**	5
2	5	10	-4.99e+00	-5.99e+00	8.425330e+00	5
3	10	15	-4.99e+00	-5.99e+00	8.425330e+00	5
4	20	25	5.99e+00	5.00e+00	8.425330e+00	5
5	50	55	3.04e+00	6.97e+00	8.425330e+00	5
6	100	108	-3.68e-04	-1.00e+00	8.425330e+00	5

Алгоритм 2

	m	**f**	**x**	**y**	**max**	**seed**
1	2	3	5.99e+00	5.00e+00	8.374528e+00	1
2	5	6	5.99e+00	5.00e+00	8.374528e+00	1
3	10	11	5.99e+00	5.00e+00	8.374528e+00	1
4	20	21	5.99e+00	5.00e+00	8.374528e+00	1
5	50	51	5.99e+00	5.00e+00	8.374528e+00	1
6	100	101	5.99e+00	5.00e+00	8.374528e+00	1
7	200	201	5.99e+00	5.00e+00	8.374528e+00	1
8	500	501	5.99e+00	5.00e+00	8.374528e+00	1
9	1000	1001	5.99e+00	5.00e+00	8.374528e+00	1
10	2000	2001	5.99e+00	5.00e+00	8.374528e+00	1
11	**5000**	**5003**	-4.19e-04	-1.00e+00	**8.425330e+00**	1
12	10000	10003	-4.19e-04	-1.00e+00	8.425330e+00	1
1	2	3	5.99e+00	5.00e+00	8.374528e+00	2

2	5	6	5.99e+00	5.00e+00	8.374528e+00	2
3	10	11	5.99e+00	5.00e+00	8.374528e+00	2
4	20	21	5.99e+00	5.00e+00	8.374528e+00	2
5	50	51	5.99e+00	5.00e+00	8.374528e+00	2
6	100	101	5.99e+00	5.00e+00	8.374528e+00	2
7	200	201	5.99e+00	5.00e+00	8.374528e+00	2
8	500	501	5.99e+00	5.00e+00	8.374528e+00	2
9	1000	1001	5.99e+00	5.00e+00	8.374528e+00	2
10	**2000**	**2003**	-3.58e-04	-1.00e+00	**8.425330e+00**	2
11	5000	5003	-3.58e-04	-1.00e+00	8.425330e+00	2
12	10000	10003	-3.58e-04	-1.00e+00	8.425330e+00	2

1	2	3	-3.94e+00	-1.77e-02	2.654063e+00	3
2	5	6	-3.94e+00	-1.77e-02	2.654063e+00	3
3	10	13	3.04e+00	6.97e+00	3.736899e+00	3
4	20	23	3.04e+00	6.97e+00	3.736899e+00	3
5	50	53	3.04e+00	6.97e+00	3.736899e+00	3
6	100	102	3.04e+00	6.97e+00	3.736899e+00	3
7	200	203	3.04e+00	6.97e+00	3.736899e+00	3
8	500	505	5.99e+00	5.00e+00	8.374528e+00	3
9	1000	1005	5.99e+00	5.00e+00	8.374528e+00	3
10	2000	2005	5.99e+00	5.00e+00	8.374528e+00	3
11	5000	5005	5.99e+00	5.00e+00	8.374528e+00	3
12	**10000**	**10007**	-3.74e-04	-1.00e+00	**8.425330e+00**	3

1	2	3	-4.99e+00	-5.99e+00	3.284624e+00	4
2	5	6	-4.99e+00	-5.99e+00	3.284624e+00	4
3	10	13	5.99e+00	5.00e+00	8.374528e+00	4
4	20	23	5.99e+00	5.00e+00	8.374528e+00	4
5	50	53	5.99e+00	5.00e+00	8.374528e+00	4
6	100	103	5.99e+00	5.00e+00	8.374528e+00	4
7	200	203	5.99e+00	5.00e+00	8.374528e+00	4
8	500	503	5.99e+00	5.00e+00	8.374528e+00	4
9	1000	1003	5.99e+00	5.00e+00	8.374528e+00	4
10	2000	2003	5.99e+00	5.00e+00	8.374528e+00	4
11	5000	5003	5.99e+00	5.00e+00	8.374528e+00	4
12	10000	10003	5.99e+00	5.00e+00	8.374528e+00	4

1	2	3	3.04e+00	6.97e+00	3.736899e+00	5
2	5	6	3.04e+00	6.97e+00	3.736899e+00	5
3	10	11	3.04e+00	6.97e+00	3.736899e+00	5
4	20	21	3.04e+00	6.97e+00	3.736899e+00	5
5	50	51	3.04e+00	6.97e+00	3.736899e+00	5

6	**100**	**103**	-4.27e-04	-1.00e+00	**8.425330e+00**	5
7	200	203	-4.27e-04	-1.00e+00	8.425330e+00	5
8	500	503	-4.27e-04	-1.00e+00	8.425330e+00	5
9	1000	1003	-4.27e-04	-1.00e+00	8.425330e+00	5
10	2000	2003	-4.27e-04	-1.00e+00	8.425330e+00	5
11	5000	5003	-4.27e-04	-1.00e+00	8.425330e+00	5
12	10000	10003	-4.27e-04	-1.00e+00	8.425330e+00	5

Алгоритм 3

	m	**f**	**x**	**y**	**max**	**seed**
1	**2**	**68**	-4.48e-04	-1.00e+00	**8.425330e+00**	1
2	5	192	-4.06e-04	-1.00e+00	8.425330e+00	1
3	10	363	-4.06e-04	-1.00e+00	8.425330e+00	1
4	20	697	-4.06e-04	-1.00e+00	8.425330e+00	1
5	50	1492	-4.06e-04	-1.00e+00	8.425330e+00	1
6	100	3121	-4.06e-04	-1.00e+00	8.425330e+00	1
1	**2**	**57**	-3.69e-04	-1.00e+00	**8.425330e+00**	2
2	5	122	-3.69e-04	-1.00e+00	8.425330e+00	2
3	10	270	-3.69e-04	-1.00e+00	8.425330e+00	2
4	20	561	-3.69e-04	-1.00e+00	8.425330e+00	2
5	50	1290	-4.18e-04	-1.00e+00	8.425330e+00	2
6	100	2697	-4.18e-04	-1.00e+00	8.425330e+00	2
1	2	65	-3.94e+00	-1.77e-02	2.654063e+00	3
2	5	215	-3.94e+00	-1.77e-02	2.654063e+00	3
3	**10**	**389**	-3.57e-04	-1.00e+00	**8.425330e+00**	3
4	20	798	-4.18e-04	-1.00e+00	8.425330e+00	3
5	50	1675	-4.18e-04	-1.00e+00	8.425330e+00	3
6	100	3381	-4.18e-04	-1.00e+00	8.425330e+00	3
1	2	57	-4.99e+00	-5.99e+00	3.284624e+00	4
2	**5**	**173**	-3.69e-04	-1.00e+00	**8.425330e+00**	4
3	10	254	-3.69e-04	-1.00e+00	8.425330e+00	4
4	20	605	-4.09e-04	-1.00e+00	8.425330e+00	4
5	50	1667	-4.09e-04	-1.00e+00	8.425330e+00	4
6	100	3375	-4.09e-04	-1.00e+00	8.425330e+00	4
1	2	48	5.99e+00	5.00e+00	8.374528e+00	5
2	**5**	**120**	-3.85e-04	-1.00e+00	**8.425330e+00**	5
3	10	265	-4.21e-04	-1.00e+00	8.425330e+00	5
4	20	480	-4.21e-04	-1.00e+00	8.425330e+00	5
5	50	1423	-4.21e-04	-1.00e+00	8.425330e+00	5
6	100	3042	-4.21e-04	-1.00e+00	8.425330e+00	5

4. Код программы

```
D = [[-10, 10], [-10, 10]]

def f(x):
    C, a, b = [2, 3, 8, 3, 2, 8], [3, -5, 0, 3, -4, 6], [-4, -6, -1, 7, 0, 5]
    f_sum = 0
    for i in range(6):
        f_sum += C[i] / (1 + (x[0] - a[i])**2 + (x[1] - b[i])**2)
    return f_sum

def f_neg(x):
    return - f(x)

def plot_func(view_init, ticks=True):
    fig = plt.figure(figsize=(14, 12))
    ax = fig.add_subplot(projection='3d')
    x = y = np.arange(-10.0, 10.0, 0.1)
    X, Y = np.meshgrid(x, y)
    zs = np.array([f([x, y]) for x, y in zip(np.ravel(X), np.ravel(Y))])
    Z = zs.reshape(X.shape)
    ax.plot_surface(X, Y, Z)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('f')
    if ticks == False: plt.axis('off')
    ax.view_init(view_init[0], view_init[1])
    plt.show()

#plot_func([20, 15])
#plot_func([90, 90], ticks=False)
```

Простой случайный поиск

```
def random_search(eps, P, seed, boundaries=D, f=f):
    random.seed(seed)
    P_eps = (eps * eps) / (D[0][1] - D[0][0]) * (D[1][1] - D[1][0])
    N = int(np.log(1.0 - P) / np.log(1.0 - P_eps) + 1)
    x_best = np.array([float()] * 2)
    for i in range(2):
        x_best[i] = random.uniform(D[i][0], D[i][1])
    for _ in range(N):
        x = np.array([float()] * 2)
        for i in range(2):
            x[i] = random.uniform(D[i][0], D[i][1])
        fx, fx_best = f(x), f(x_best)
        if fx > fx_best:
            fx_best, x_best = fx, x
    df = pd.DataFrame([eps, P, N, "{:.2e}".format(x_best[0]), "{:.2e}".format(x_best[1]),
"{:.6e}".format(fx_best)]).T
    df.rename(columns={0: 'eps', 1: 'P', 2: 'N', 3: 'x', 4: 'y', 5: 'max'}, inplace=True)
    return df
```

Алгоритм 1

```
def random_descents_search(m, seed, boundaries=D, f=f):
    random.seed(seed)
```

```
    n = f_count = 0
    x = np.array([float()] * 2)
    for i in range(2):
        x[i] = random.uniform(D[i][0], D[i][1])
    x_extremum = fmin(f_neg, x, disp=False)
    fx_best = f(x_extremum)
    f_count += 1
    while n != m:
        for i in range(2):
            x[i] = random.uniform(D[i][0], D[i][1])
        x_extremum = fmin(f_neg, x, disp=False)
        fx_extremum = f(x_extremum)
        f_count += 1
        if fx_extremum > fx_best:
            fx_best = fx_extremum
        else:
            n += 1
    df = pd.DataFrame([m, f_count, "{:.2e}".format(x_extremum[0]),
"{:.2e}".format(x_extremum[1]), "{:.6e}".format(fx_best), seed]).T
    df.rename(columns={0: 'm', 1: 'f', 2: 'x', 3: 'y', 4: 'max', 5: 'seed'},
inplace=True)
    return df
```

Алгоритм 2

```
def stochastic_search(m, seed, boundaries=D, f=f):
    random.seed(seed)
    n = f_count = 0
    x = np.array([float()] * 2)
    for i in range(2):
        x[i] = random.uniform(D[i][0], D[i][1])
    x_extremum = fmin(f_neg, x, disp=False)
    fx_best = f(x_extremum)
    f_count += 1
    while n != m:
        for i in range(2):
            x[i] = random.uniform(D[i][0], D[i][1])
        if f(x) > fx_best:
            x_extremum = fmin(f_neg, x, disp=False)
            fx_best = f(x_extremum)
        else:
            n += 1
        f_count += 1
    df = pd.DataFrame([m, f_count, "{:.2e}".format(x_extremum[0]),
"{:.2e}".format(x_extremum[1]), "{:.6e}".format(fx_best), seed]).T
    df.rename(columns={0: 'm', 1: 'f', 2: 'x', 3: 'y', 4: 'max', 5: 'seed'},
inplace=True)
    return df
```

Алгоритм 3

```
def random_direction_search(m, seed, boundaries=D, f=f):
    random.seed(seed)
    n = f_count = 0
    x = random_step = np.array([float()] * 2)
    for i in range(2):
```

```
        x[i] = random.uniform(D[i][0], D[i][1])
    x_extremum = fmin(f_neg, x, disp=False)
    fx_best = f(x_extremum)
    f_count += 1
    while n != m:
        for i in range(2):
            random_step[i] = random.choice([-0.5, 0.5, -0.4, 0.4, -0.3, 0.3, -0.2, 0.2])
        for i in range(1, 100):
            x_prev = x_extremum - random_step * (i-1)
            x_new = x_extremum - random_step * i
            if (x_new[0] > 10) or (x_new[1] > 10) or (x_new[0] < -10) or (x_new[1] <
-10):
                break
            slope = f(x_new) - f(x_prev)
            f_count += 2
            if slope > 0.001:
                x_extremum_new = fmin(f_neg, x_new, disp=False)
                fx_extremum_new = f(x_extremum_new)
                f_count += 1
                break
        if slope < 0.001:
            n += 1
        elif fx_extremum_new > fx_best:
            fx_best = fx_extremum_new
            x_extremum = x_extremum_new
        else:
            n += 1
    df = pd.DataFrame([m, f_count, "{:.2e}".format(x_extremum[0]),
"{:.2e}".format(x_extremum[1]), "{:.6e}".format(fx_best), seed]).T
    df.rename(columns={0: 'm', 1: 'f', 2: 'x', 3: 'y', 4: 'max', 5: 'seed'},
inplace=True)
    return df
```

5. Выводы

В первом и третьем алгоритмах глобального поиска для нахождения экстремума функции потребовалось примерно равное значение m при разных начальных значениях ГСЧ (2-5 в первом, 2-10 в третьем).
Эффективность данных алгоритмов заметна на исследуемой функции, поскольку она имеет всего 6 хорошо выраженных экстремумов на довольно ограниченной области.

Второй алгоритм может быть эффективен в том случае, когда количество экстремумов велико, а область исследования достаточно большая. Это связано с тем, что он не вычисляет целевую функцию для спуска в локальный экстремум, а сравнивает значение случайно выбранной на области точки со значением уже полученного ранее экстремума. Спуск в локальный экстремум происходит только в том случае, если значение функции в новой точке окажется лучше значения функции в экстремуме.

Первый алгоритм находит глобальный экстремум довольно быстро, но при большом количестве экстремумов на области будет вынужден совершать бессмысленные

вычисления целевой функции. Это связано с тем, что он сравнивает значение функции только после спуска в локальный экстремум.

Третий алгоритм может исправить проблему большого количества экстремумов, однако он более уязвим к увеличению области исследования. В случае, если экстремумы находятся на достаточно большом расстоянии друг от друга, данный алгоритм будет довольно долго выходить из области притяжения очередного экстремума, что отразится на количестве вычислений целевой функции.

Таким образом, первый и третий алгоритмы являются более устойчивыми, чем второй алгоритм, но и более трудоемкими на больших и сложных областях исследования.

Простой случайный поиск возможно использовать только в том случае, когда область исследования очень мала. В нашем случае потребовалось несколько десятков тысяч испытаний, чтобы найти глобальный экстремум с высокой точностью. И поскольку целевая функция вычисляется при каждом испытании, этот метод является в разы менее эффективным, чем его модификации.