МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Московский государственный технический университет

им. Н.Э. Баумана

(МГТУ им. Н.Э. Баумана)

Кафедра «Информационная безопасность» (ИУ8)

Лабораторная работа № 2

Исследование методов прямого поиска экстремума унимодальной функции одного переменного

Выполнила: Броцкий К.А.

студент группы ИУ8-32

Проверила: Коннова Н.С.,

доцент каф. ИУ8

г. Москва 2020 г.

Цель работы

Изучение метода случайного поиска экстремума на примере унимодальной и мультимодальной функций одного переменного.

Постановка задачи

1. На интервале $[a,b]$ задана унимодальная функция одного переменного $f(x)$. Используя метод *случайного поиска* осуществить поиск минимума $f(x)$ с заданной вероятностью попадания в окрестность экстремума P при допустимой длине интервала неопределенности ε. Определить необходимое число испытаний *N*. Численный эксперимент выполнить для значений $P = 0,90, 0,91, ..., 0,99$ и значений $\varepsilon = (b-a)q$, где $q = 0,005, 0,010, ..., 0,100$.

Последовательность действий:

- определить вероятность P_1 непопадания в ε-окрестность экстремума за одной испытание;

- записать выражение для вероятности P_N непопадания в ε-окрестность экстремума за *N* испытаний;

- из выражения для P_N определить необходимое число испытаний *N* в зависимости от заданных $P_N = P$ и ε.

2. При аналогичных исходных условиях осуществить поиск минимума $f(x)$, модулированной сигналом $\sin 5x$, т.е. *мультимодальной* функции $f(x)\cdot\sin 5x$.

Вариант 1:

№пп	Функция $f(x)$	a	b
1	$-0,5\cos 0,5x - 0,5$	–5	2

График функции

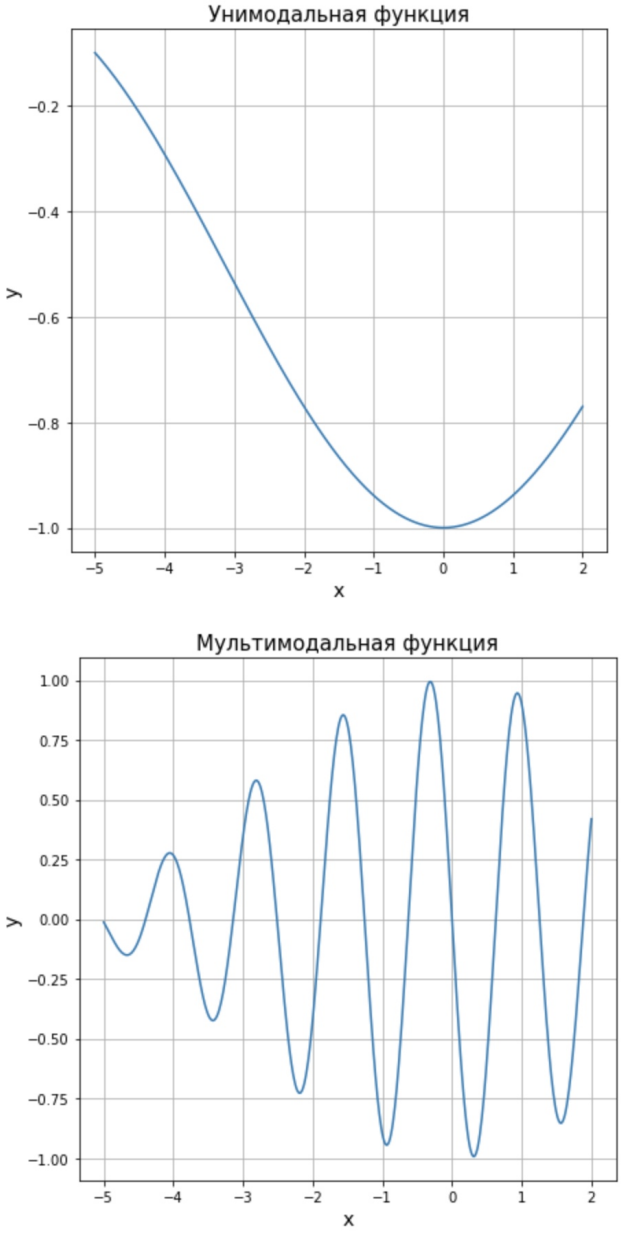

Рис.1. Графики функций

Скриншот консоли

q\P	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
0.005	460	481	504	531	562	598	643	700	781	919
0.01	230	240	252	265	280	299	321	349	390	459
0.015	153	160	168	176	187	199	213	233	259	305
0.02	114	120	126	132	140	149	160	174	194	228
0.025	91	96	100	106	112	119	128	139	155	182
0.03	76	80	83	88	93	99	106	116	129	152
0.035	65	68	71	75	79	85	91	99	110	130
0.04	57	59	62	66	69	74	79	86	96	113
0.045	51	53	55	58	62	66	70	77	85	101
0.05	45	47	50	52	55	59	63	69	77	90
0.055	41	43	45	48	50	53	57	62	70	82
0.06	38	39	41	43	46	49	53	57	64	75
0.065	35	36	38	40	42	45	48	53	59	69
0.07	32	34	35	37	39	42	45	49	54	64
0.075	30	31	33	35	37	39	42	45	51	60
0.08	28	29	31	32	34	36	39	43	47	56
0.085	26	28	29	30	32	34	37	40	45	52
0.09	25	26	27	29	30	32	35	38	42	49
0.095	24	25	26	27	29	31	33	36	40	47
0.1	22	23	24	26	27	29	31	34	38	44

q\P	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
0.005	-0.999999	-0.999999	-0.999978	-0.999996	-1	-0.999961	-0.999999	-1	-1	-0.999999
0.01	-0.999979	-0.999962	-0.999986	-0.999985	-0.999996	-0.999961	-1	-0.999999	-1	-1
0.015	-0.999961	-0.999473	-0.999998	-1	-0.999977	-0.999982	-0.999971	-0.999998	-1	-1
0.02	-0.999592	-0.999999	-0.999866	-0.999974	-0.999999	-1	-0.999996	-0.999656	-1	-0.999971
0.025	-0.999977	-0.999916	-0.99998	-0.999878	-0.999924	-0.999821	-0.999997	-0.99985	-0.999927	-0.99981
0.03	-0.998598	-0.999963	-0.999993	-1	-0.999831	-1	-0.999201	-0.999995	-0.999995	-0.999938
0.035	-0.999574	-0.999978	-0.999406	-1	-0.999974	-0.999782	-0.999441	-0.999995	-0.999645	-0.999994
0.04	-0.999973	-0.999667	-0.998028	-0.999996	-0.999831	-1	-0.999972	-1	-0.999997	-0.999988
0.045	-0.997368	-0.999076	-0.999999	-0.998861	-0.997015	-0.999796	-0.999652	-0.999847	-0.999919	-1
0.05	-1	-0.999999	-0.999991	-0.999906	-0.998724	-0.999928	-0.999691	-0.999986	-0.999967	-0.999998
0.055	-0.999694	-0.999984	-0.998467	-0.999982	-0.998301	-0.999362	-0.999627	-0.999211	-0.999828	-0.999989
0.06	-0.999957	-0.999649	-0.998483	-0.999672	-0.999959	-0.999999	-0.998989	-0.999923	-0.999684	-0.999875
0.065	-0.999293	-0.999673	-0.999532	-0.996559	-0.999777	-0.999359	-0.999225	-0.996404	-0.999955	-0.999999
0.07	-0.999985	-0.99965	-0.999868	-0.999587	-0.999999	-0.999179	-0.996193	-0.999946	-0.999981	-0.999824
0.075	-0.999266	-0.999523	-0.999939	-0.999376	-0.987311	-0.999747	-0.999981	-0.999872	-0.999935	-0.99998
0.08	-0.999761	-0.999882	-0.996954	-0.999997	-0.991357	-0.999796	-0.998337	-0.999992	-0.999565	-0.999974
0.085	-0.999021	-0.999544	-0.99981	-0.998839	-0.998394	-0.999753	-0.99944	-0.999951	-0.999401	-0.999999
0.09	-0.999893	-0.99283	-0.998168	-0.999998	-0.999998	-0.999995	-0.99982	-0.999345	-0.998121	-0.999562
0.095	-0.999637	-0.999726	-0.999644	-0.999808	-0.999487	-0.999477	-0.999998	-0.999727	-0.99707	-0.999993
0.1	-0.998628	-0.988386	-0.999855	-0.999781	-0.994893	-0.988418	-0.997058	-0.999344	-0.998954	-0.999554

q\P	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
0.005	-0.993525	-0.991539	-0.993517	-0.993538	-0.989571	-0.99367	-0.993794	-0.993161	-0.993853	-0.99387
0.01	-0.988585	-0.992425	-0.989679	-0.990471	-0.988492	-0.993866	-0.993799	-0.992549	-0.993843	-0.993828
0.015	-0.94511	-0.993553	-0.988897	-0.98796	-0.991443	-0.993342	-0.993832	-0.99365	-0.992959	-0.99304
0.02	-0.989884	-0.945377	-0.993357	-0.944197	-0.991983	-0.993694	-0.982121	-0.973631	-0.993746	-0.991268
0.025	-0.992349	-0.991142	-0.99384	-0.983448	-0.992799	-0.99007	-0.961025	-0.969959	-0.991774	-0.985896
0.03	-0.993873	-0.989254	-0.985474	-0.99007	-0.941726	-0.989984	-0.966928	-0.993165	-0.978354	-0.99382
0.035	-0.992141	-0.949922	-0.987662	-0.97358	-0.934089	-0.990252	-0.992276	-0.983143	-0.943584	-0.971809
0.04	-0.942732	-0.941615	-0.957384	-0.974089	-0.986985	-0.986672	-0.99125	-0.991011	-0.976205	-0.988122
0.045	-0.976602	-0.958755	-0.852354	-0.871143	-0.955672	-0.993842	-0.982005	-0.930525	-0.985312	-0.990362
0.05	-0.992529	-0.958405	-0.977753	-0.967287	-0.987616	-0.986969	-0.993394	-0.962881	-0.991065	-0.938638
0.055	-0.93166	-0.989921	-0.963071	-0.992808	-0.993377	-0.942874	-0.935662	-0.917602	-0.945501	-0.993589
0.06	-0.993369	-0.928832	-0.841918	-0.937316	-0.99148	-0.979109	-0.92954	-0.982323	-0.991828	-0.942039
0.065	-0.976014	-0.883087	-0.974102	-0.943997	-0.987009	-0.955733	-0.938384	-0.989819	-0.962911	-0.945106
0.07	-0.94518	-0.926378	-0.989676	-0.899677	-0.941672	-0.993872	-0.990361	-0.993828	-0.992525	-0.993868
0.075	-0.875656	-0.943286	-0.836089	-0.799032	-0.910696	-0.993189	-0.992131	-0.868117	-0.944945	-0.977078
0.08	-0.99377	-0.953081	-0.970747	-0.992178	-0.850039	-0.952114	-0.983414	-0.944721	-0.905711	-0.987346
0.085	-0.945574	-0.780577	-0.842677	-0.918468	-0.992863	-0.728334	-0.98005	-0.944689	-0.936751	-0.85612
0.09	-0.993133	-0.98741	-0.993862	-0.981887	-0.984636	-0.917996	-0.945468	-0.919609	-0.945724	-0.944973
0.095	-0.937384	-0.909247	-0.914758	-0.939455	-0.888536	-0.889653	-0.808183	-0.992412	-0.942376	-0.95221
0.1	-0.685574	-0.938296	-0.926472	-0.962845	-0.945584	-0.99261	-0.971286	-0.934615	-0.99111	-0.989686

Рис.2. Скриншот консоли

Листинг программы с реализацией алгоритмов на C++

```
#include <iostream>
#include <cmath>
#include <vector>
#include <string>
#include <iomanip>

double f_of_x(const double x) {
    return ((-0.5) * std::cos(0.5 * x) - 0.5);
}

double f_of_new_x(const double x) {
    return (f_of_x(x) * sin(5 * x));
}

const std::vector<double> P = {0.9, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96,
0.97, 0.98, 0.99},
```

```
        q = {0.005, 0.01, 0.015, 0.02, 0.025, 0.03, 0.035, 0.04, 0.045, 0.05,
0.055, 0.06, 0.065, 0.07, 0.075, 0.08,
             0.085, 0.09, 0.095, 0.1};

const double a = -5.0, b = 2.0;


double random(double min, double max) {
    return (double) (rand()) / RAND_MAX * (max - min) + min;
}

std::vector<std::vector<int>> n_p_of_q(const std::vector<double> &P,
const std::vector<double> &q) {
    std::vector<std::vector<int>> table;
    for (size_t i = 0; i < q.size(); i++) {
        std::vector<int> string;
        for (size_t j = 0; j < P.size(); j++) {
            string.push_back(std::ceil(log(1 - P[j]) / log(1 - q[i]) ));
        }
        table.push_back(string);
    }
    return table;
}

void pr_n_p_of_q(const std::vector<std::vector<int>> &table) {
    std::cout << std::string(68, '-') << std::endl;
    std::cout << "|q\\\P    ";
    for (size_t i = 0; i < P.size(); i++) {
        std::cout << "|" << std::setw(5) << std::left << P[i];
    }
    std::cout << '|' << std::endl;
    std::cout << std::string(68, '-') << std::endl;
    for (size_t i = 0; i < table.size(); i++) {
        std::cout << "|";
        std::cout << std::setw(6) << std::left << q[i];
        for (size_t j = 0; j < table[i].size(); j++) {
            std::cout << '|' << std::setw(5) << std::left << table[i][j];
        }
        std::cout << '|' << std::endl;
    }
    std::cout << std::string(68, '-') << std::endl;
}

void print_table(const std::vector<std::vector<double>> &table) {
    std::cout << std::string(118, '-') << std::endl;
```

```
    std::cout << "|q\\\P   ";
    for (size_t i = 0; i < P.size(); i++) {
        std::cout << "|" << std::setw(10) << std::left << P[i];
    }
    std::cout << '|' << std::endl;
    std::cout << std::string(118, '-') << std::endl;
    for (size_t i = 0; i < table.size(); i++) {
        std::cout << "|";
        std::cout << std::setw(6) << std::left << q[i];
        for (size_t j = 0; j < table[i].size(); j++) {
            std::cout << '|' << std::setw(10) << std::left << table[i][j];
        }
        std::cout << '|' << std::endl;
    }
    std::cout << std::string(118, '-') << std::endl;
}

std::vector<std::vector<double>> rand_search(const
std::vector<std::vector<int>> &all_n, const int choice) {
    std::vector<std::vector<double>> table;
    for (size_t i = 0; i < q.size(); i++) {
        std::vector<double> string;
        for (size_t j = 0; j < P.size(); j++) {
            double min = 92233720368547758O7.0;
            for (size_t k = 0; k < all_n[i][j]; k++) {
                double elem;
                if (choice == 0) {
                    elem = f_of_x(random(a, b));
                } else if (choice == 1) {
                    elem = f_of_new_x(random(a, b));
                } else {
                    throw std::logic_error("Invalid choice");
                }
                if (elem < min) {
                    min = elem;
                }
            }
            string.push_back(min);
        }
        table.push_back(string);
    }
    return table;
}

int main() {
```

```
    pr_n_p_of_q(n_p_of_q(P, q));
    std::cout << std::endl;

    print_table(rand_search(n_p_of_q(P, q), 0));
    std::cout << std::endl;

    print_table(rand_search(n_p_of_q(P, q), 1));
    std::cout << std::endl;

    return 0;
}
```

Контрольный вопрос

В чем состоит сущность метода случайного поиска? Какова область применимости данного метода?

При таком поиске все последующие испытания проводят совершенно независимо от результатов предыдущих. Сходимость такого поиска очень мала, но имеется важное преимущество, связанное с возможностью решения многоэкстремальных задач (искать глобальный экстремум). Примером ненаправленного поиска является рассмотренный простой случайный поиск.

Вывод

Таким образом, в результате вычисления минимума унимодальной на данном отрезке функции различными методами, мы убедились в том, что количество итераций для метода золотого сечения меньше, чем для метода оптимального пассивного поиска, следовательно, он эффективнее.