Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

Отчёт

по лабораторной работе № 2
по дисциплине «Теория систем и системный анализ»

Тема: «Исследование метода случайного поиска экстремума функции одного переменного**»**

Вариант 3

Выполнила: Бакаев Ф.Б., студент группы ИУ8-31

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.

Условие задачи

1. На интервале [0,3] a b задана унимодальная функция одного переменного f(x) = -sqrt(x)*sin(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 ε = (b – a) q , где q = 0,005, 0,010,..., 0,100.

Последовательность действий:

- определить вероятность P1 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) * sin5(x).

Графики заданных функциий

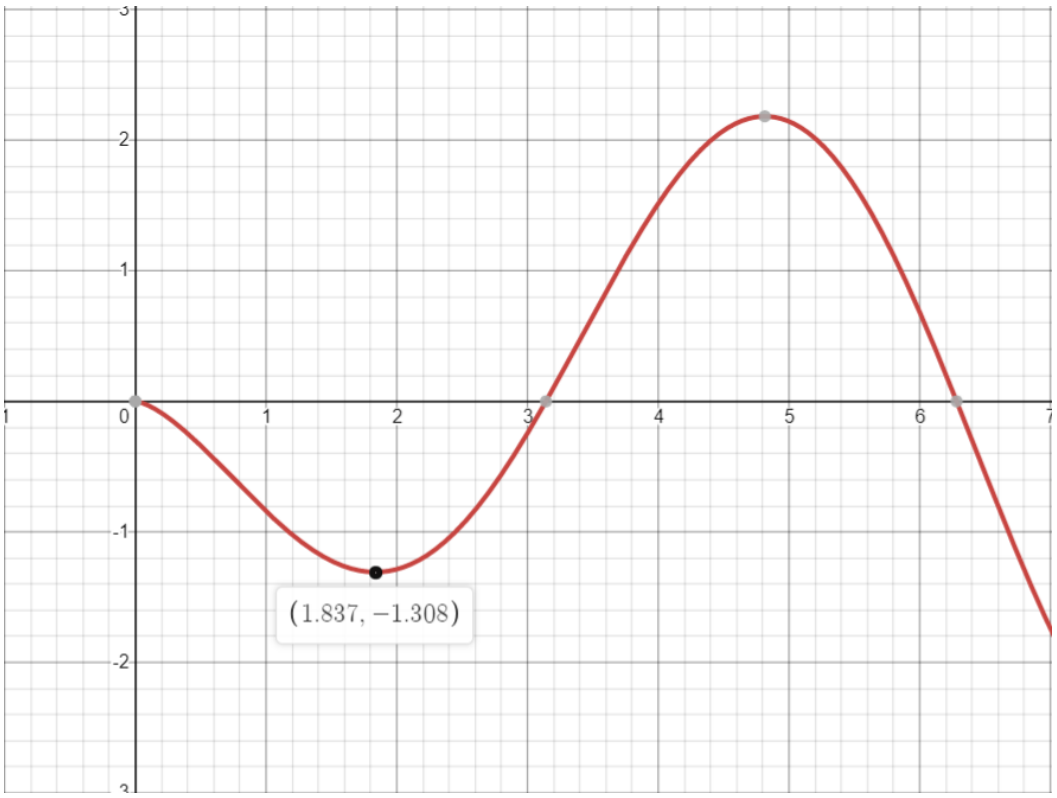

Рисунок 1 - График функции y = -sqrt(x) * sin(x) на [0;3]

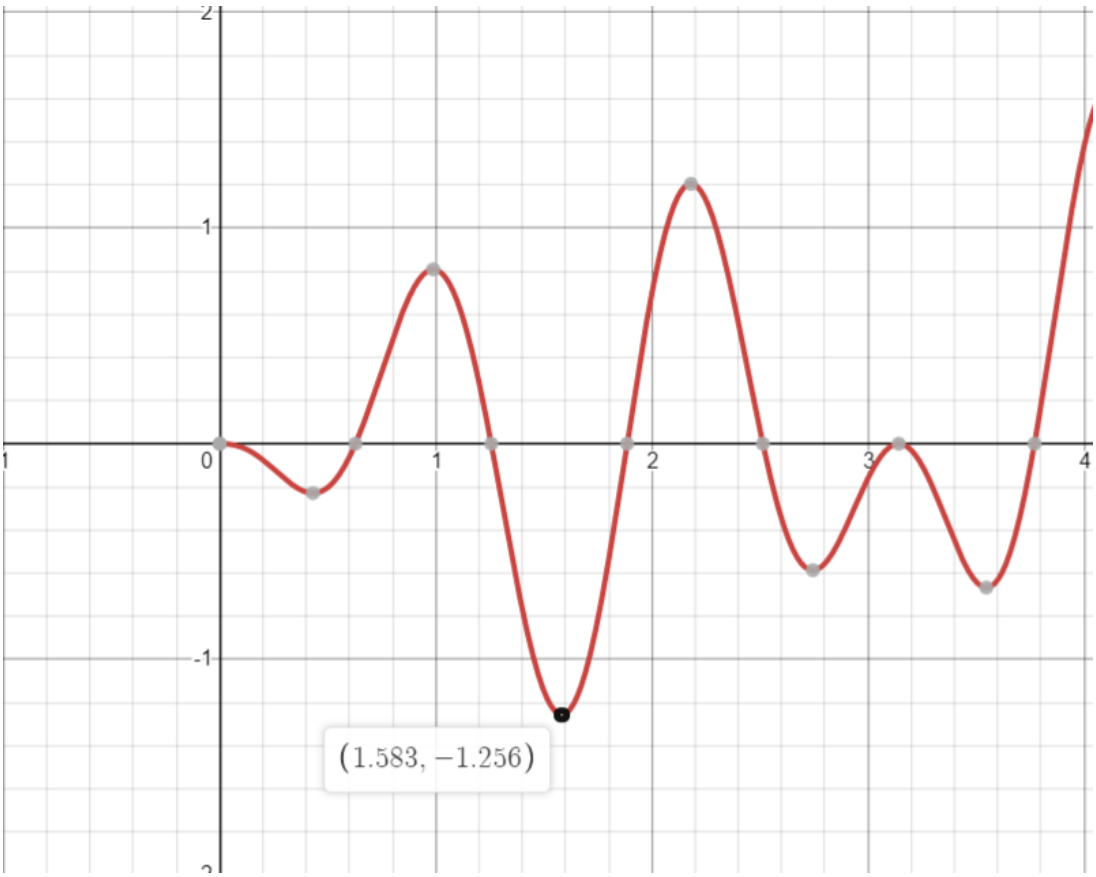

Рисунок 2 - График функции y = -sqrt(x) * sin(x) * sin(5x) на [0;3]

Зависимость N от p и q

q\P	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
0.005	459	480	503	530	561	597	642	699	780	918
0.01	229	239	251	264	279	298	320	348	389	458
0.015	152	159	167	175	186	198	212	232	258	304
0.02	113	119	125	131	139	148	159	173	193	227
0.025	90	95	99	105	111	118	127	138	154	181
0.03	75	79	82	87	92	98	105	115	128	151
0.035	64	67	70	74	78	84	90	98	109	129
0.04	56	58	61	65	68	73	78	85	95	112
0.045	50	52	54	57	61	65	69	76	84	100
0.05	44	46	49	51	54	58	62	68	76	89
0.055	40	42	44	47	49	52	56	61	69	81
0.06	37	38	40	42	45	48	52	56	63	74
0.065	34	35	37	39	41	44	47	52	58	68
0.07	31	33	34	36	38	41	44	48	53	63
0.075	29	30	32	34	36	38	41	44	50	59
0.08	27	28	30	31	33	35	38	42	46	55
0.085	25	27	28	29	31	33	36	39	44	51
0.09	24	25	26	28	29	31	34	37	41	48
0.095	23	24	25	26	28	30	32	35	39	46
0.1	21	22	23	25	26	28	30	33	37	43

Рисунок 3 – Таблица зависимости N от p и q

Случайный поиск для заданных функций

q\P	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
0.005	-1.30762	-1.30762	-1.30756	-1.30762	-1.30762	-1.30758	-1.30762	-1.30762	-1.30762	-1.30762
0.01	-1.30762	-1.30749	-1.30758	-1.30762	-1.30762	-1.30758	-1.30755	-1.30762	-1.30762	-1.30762
0.015	-1.30758	-1.30762	-1.30738	-1.30751	-1.30762	-1.30759	-1.3076	-1.3076	-1.30757	-1.3076
0.02	-1.30762	-1.30761	-1.30755	-1.30761	-1.30753	-1.30761	-1.30755	-1.30762	-1.30761	-1.30762
0.025	-1.30762	-1.30762	-1.30742	-1.30738	-1.30761	-1.30759	-1.30761	-1.30657	-1.30755	-1.30761
0.03	-1.30748	-1.307	-1.30734	-1.30352	-1.30758	-1.30761	-1.30741	-1.30761	-1.30759	-1.30756
0.035	-1.30748	-1.30753	-1.30759	-1.30753	-1.3076	-1.30745	-1.30762	-1.3075	-1.30762	-1.30655
0.04	-1.30747	-1.3076	-1.30675	-1.30761	-1.30106	-1.30622	-1.30759	-1.30749	-1.30518	-1.30747
0.045	-1.30656	-1.30762	-1.30756	-1.30533	-1.307	-1.30761	-1.30744	-1.30762	-1.30759	-1.307
0.05	-1.30753	-1.30672	-1.30694	-1.30737	-1.30743	-1.30742	-1.30727	-1.3076	-1.30568	-1.30752
0.055	-1.30707	-1.30728	-1.30762	-1.30397	-1.30761	-1.30759	-1.3064	-1.30759	-1.30588	-1.30753
0.06	-1.30676	-1.30705	-1.30761	-1.3076	-1.30495	-1.30671	-1.30714	-1.3076	-1.30727	-1.30761
0.065	-1.29991	-1.30719	-1.30558	-1.30742	-1.30762	-1.30661	-1.30329	-1.30656	-1.30504	-1.30714
0.07	-1.30758	-1.30565	-1.30747	-1.30644	-1.2928	-1.30759	-1.30759	-1.30711	-1.30621	-1.30759
0.075	-1.305	-1.3071	-1.30754	-1.30494	-1.29698	-1.30657	-1.30751	-1.30747	-1.30045	-1.30714
0.08	-1.30655	-1.30752	-1.30544	-1.30571	-1.3065	-1.3069	-1.30432	-1.30758	-1.30759	-1.30725
0.085	-1.30645	-1.30761	-1.30688	-1.30377	-1.30671	-1.3076	-1.30751	-1.30625	-1.30762	-1.30732
0.09	-1.29622	-1.30757	-1.30762	-1.30735	-1.30414	-1.30759	-1.30257	-1.30733	-1.30733	-1.30677
0.095	-1.30715	-1.30112	-1.30654	-1.30706	-1.30496	-1.2771	-1.29811	-1.30715	-1.30752	-1.30633
0.1	-1.30694	-1.299	-1.3047	-1.30471	-1.30304	-1.30743	-1.30418	-1.29653	-1.30718	-1.3075

Рисунок 4 – Случайный поиск для y = -sqrt(x) * sin(x) на [0;3]

q\P	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
0.005	-1.25574	-1.25568	-1.25573	-1.25573	-1.25464	-1.25554	-1.25573	-1.2557	-1.25547	-1.25574
0.01	-1.25543	-1.2426	-1.25518	-1.25558	-1.23813	-1.25558	-1.25515	-1.25573	-1.25477	-1.25564
0.015	-1.25299	-1.25416	-1.25258	-1.25544	-1.25549	-1.25489	-1.25507	-1.25317	-1.25574	-1.25543
0.02	-1.25258	-1.25513	-1.25572	-1.2555	-1.24666	-1.25043	-1.25367	-1.25465	-1.25538	-1.25565
0.025	-1.2555	-1.25418	-1.2477	-1.25511	-1.25547	-1.2531	-1.25571	-1.25406	-1.25572	-1.25032
0.03	-1.25574	-1.24376	-1.24092	-1.25544	-1.25443	-1.25554	-1.25192	-1.22527	-1.25573	-1.25529
0.035	-1.15846	-1.25515	-1.2555	-1.25188	-1.25438	-1.25452	-1.25565	-1.25229	-1.25572	-1.25571
0.04	-1.23655	-1.23344	-1.25542	-1.25442	-1.18574	-1.25566	-1.25499	-1.25514	-1.2463	-1.25573
0.045	-1.20683	-1.10964	-1.20367	-1.24776	-1.25551	-1.25111	-1.25546	-1.24743	-1.22766	-1.25048
0.05	-1.24659	-1.25416	-1.25567	-1.24586	-1.25573	-1.24818	-1.24293	-1.24792	-1.25443	-1.24906
0.055	-1.11017	-1.25527	-1.25237	-1.20683	-1.14723	-1.25569	-1.25534	-1.24083	-1.24726	-1.25358
0.06	-1.20781	-1.1781	-1.25554	-1.25554	-1.24987	-1.25404	-1.25498	-1.22231	-1.25322	-1.25419
0.065	-0.585952	-1.25573	-1.09074	-1.20963	-1.13769	-1.23006	-1.22911	-1.25332	-1.23093	-1.2545
0.07	-1.23747	-1.25545	-0.867882	-1.25259	-1.25303	-1.14581	-1.18317	-1.25571	-1.14095	-1.25528
0.075	-1.16546	-1.1991	-1.15637	-1.25027	-1.21575	-1.2547	-1.25271	-1.23561	-1.25559	-1.25233
0.08	-1.24628	-1.23213	-1.25095	-1.25315	-1.25069	-1.23603	-1.2468	-1.17997	-1.23686	-1.23444
0.085	-1.02069	-1.15048	-1.23571	-1.24822	-1.24666	-1.24393	-1.24	-0.744016	-1.22715	-1.2543
0.09	-1.24993	-1.25568	-1.25457	-1.21635	-1.25267	-1.24947	-1.25547	-1.25524	-1.2557	-1.21641
0.095	-1.16854	-1.24623	-1.25064	-0.93822	-1.24058	-1.23833	-1.25572	-1.22899	-1.24478	-1.23487
0.1	-1.13983	-0.934013	-1.24046	-1.16068	-1.21798	-1.08995	-1.24763	-1.24992	-1.25425	-1.24958

Рисунок 5 – Случайный поиск для y = -sqrt(x) * sin(x) * sin(5x) на [0;3]

Выводы

Из полученных таблиц и графиков видно, что применимость метода случайного поиска не зависит от того, является ли функция унимодальной или мультимодальной. Для увеличения вероятности попадания в заданный интервал или для уменьшения интервала неопределенности необходимо увеличивать число случайных точек.

Приложение. Исходный код программы

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

using std::cout;
using std::endl;

double f(const double x) {
    return (-sqrt(x) * sin(x));
}

double f_new(const double x) {
    return (f(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
};

const std::vector<double> 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 = 0.0;

const double b = 3.0;

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

std::vector<std::vector<int>> n_p_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(int(log(1 - P[j]) / log(1 - q[i]) * 100) / 100);
        }
        table.push_back(string);
    }
    return table;
}

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

```
        for (size_t j = 0; j < table[i].size(); j++) {
            cout << '|' << std::setw(5) << std::left << table[i][j];
        }
        cout << '|' << endl;
    }
    cout << std::string(68, '-') << endl;
}

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

std::vector<std::vector<double>> random_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 = 9223372036854775807.0;
            for (size_t k = 0; k < all_n[i][j]; k++) {
                double elem;
                if (choice == 0) {
                    elem = f(random(a, b));
                }
                else if (choice == 1) {
                    elem = f_new(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() {
    Print_n_p_q(n_p_q(P, q));
    cout << endl;

    Print(random_search(n_p_q(P, q), 0));
    cout << endl;

    Print(random_search(n_p_q(P, q), 1));
    cout << endl;
```

```
    return 0;
}
```