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 1

Выполнил: Александроов А.А.,
студент группы ИУ8-31
Проверила: Коннова Н.С., доцент
каф. ИУ8

г. Москва, 2020

Цель работы

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

Условие задачи

1. На интервале [-5,2] задана унимодальная функция одного переменного $f(x) = -0.5 * \sin(0.5x) - 0.5$. 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 функций

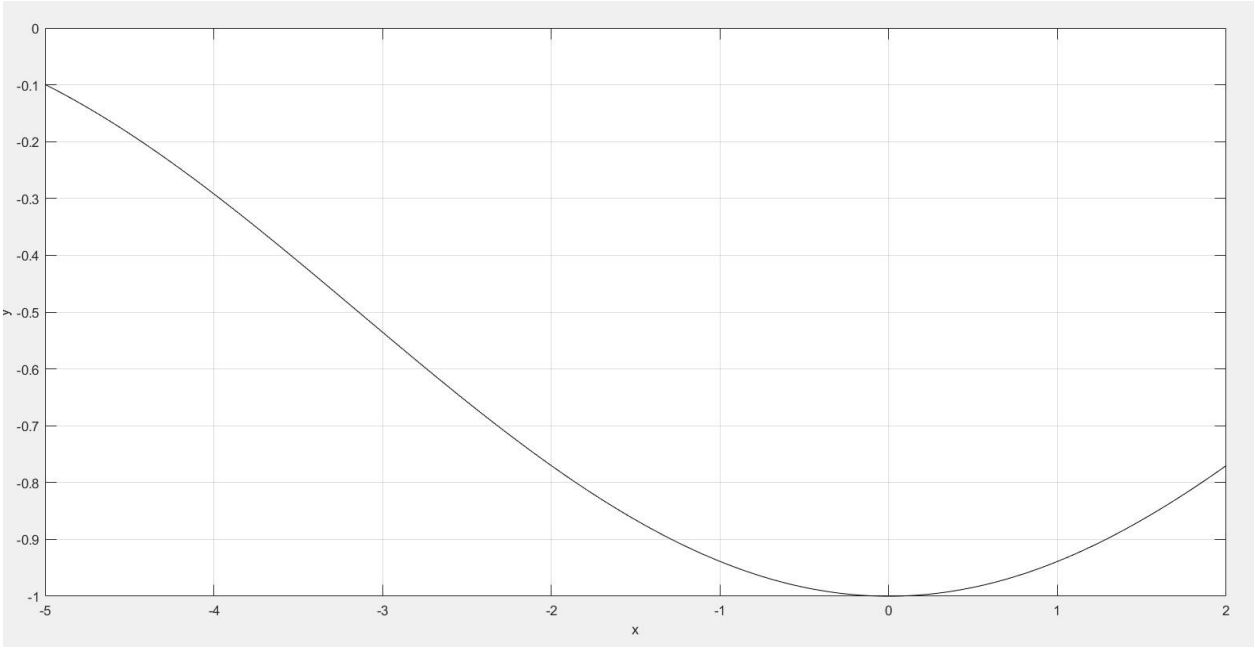

Рисунок 1 - График функции $f(x) = -0.5 * cos(0.5x) - 0.5$ на [-5;2]

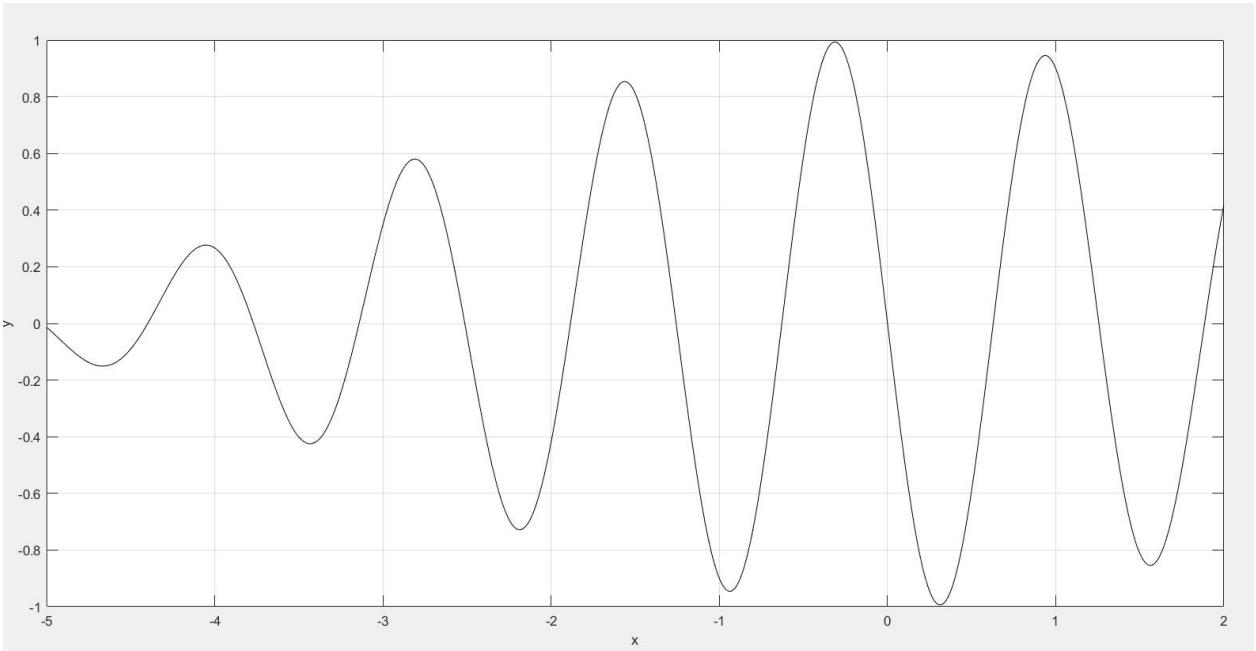

Рисунок 2 - График функции $f(x) = (-0.5 * \cos(0.5x) - 0.5) * \sin(5x)$ на [-5;2]

Зависимость N от p и q

N PRINT

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

Рисунок 3 – Таблица зависимости N от p и q

Случайный поиск для заданных функций

UNOMODAL PRINT

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	-1	-1	-1	-1	-1	-1	-1	-1	-1
0.01	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0.015	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0.02	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0.025	-0.999	-1	-1	-1	-1	-1	-1	-1	-1	-1
0.03	-1	-1	-0.999	-1	-0.999	-1	-1	-1	-1	-1
0.035	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0.04	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0.045	-1	-0.999	-1	-1	-1	-1	-1	-1	-1	-1
0.05	-0.994	-1	-1	-1	-0.999	-1	-0.998	-0.998	-1	-1
0.055	-1	-1	-1	-0.999	-0.999	-1	-1	-1	-1	-1
0.06	-1	-1	-1	-1	-0.998	-1	-0.999	-1	-1	-1
0.065	-0.998	-1	-0.999	-1	-0.998	-0.999	-1	-1	-1	-0.999
0.07	-1	-1	-1	-1	-1	-1	-0.999	-1	-1	-1
0.075	-1	-0.999	-0.999	-0.997	-1	-0.998	-1	-1	-1	-1
0.08	-0.979	-0.999	-0.999	-1	-1	-1	-1	-0.998	-1	-0.999
0.085	-1	-1	-0.998	-0.998	-1	-1	-1	-0.997	-1	-1
0.09	-0.991	-0.991	-0.995	-1	-0.996	-1	-1	-0.999	-1	-1
0.095	-0.997	-1	-0.993	-1	-0.999	-1	-1	-1	-1	-1
0.1	-0.999	-0.999	-1	-0.993	-1	-0.997	-0.999	-0.998	-1	-1

Рисунок 4 – Случайный поиск для f(x)=-0.5*cos(0.5x)-0.5 на [-5;2]

MULTIMODAL PRINT

q\p	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
0.005	-0.994	-0.994	-0.994	-0.994	-0.988	-0.993	-0.993	-0.994	-0.992	-0.994
0.01	-0.994	-0.991	-0.993	-0.982	-0.984	-0.994	-0.985	-0.984	-0.993	-0.992
0.015	-0.975	-0.99	-0.989	-0.993	-0.993	-0.992	-0.994	-0.99	-0.994	-0.994
0.02	-0.884	-0.993	-0.991	-0.898	-0.994	-0.968	-0.993	-0.988	-0.99	-0.983
0.025	-0.934	-0.923	-0.988	-0.994	-0.897	-0.944	-0.992	-0.982	-0.989	-0.994
0.03	-0.944	-0.992	-0.982	-0.994	-0.988	-0.989	-0.994	-0.945	-0.992	-0.992
0.035	-0.946	-0.991	-0.965	-0.993	-0.991	-0.933	-0.992	-0.963	-0.946	-0.993
0.04	-0.984	-0.991	-0.988	-0.941	-0.959	-0.946	-0.92	-0.965	-0.946	-0.993
0.045	-0.944	-0.949	-0.973	-0.965	-0.96	-0.959	-0.994	-0.872	-0.994	-0.989
0.05	-0.868	-0.994	-0.931	-0.985	-0.98	-0.941	-0.957	-0.994	-0.939	-0.943
0.055	-0.968	-0.994	-0.992	-0.921	-0.993	-0.934	-0.926	-0.992	-0.946	-0.942
0.06	-0.993	-0.975	-0.945	-0.852	-0.83	-0.974	-0.972	-0.958	-0.963	-0.994
0.065	-0.945	-0.972	-0.814	-0.965	-0.987	-0.986	-0.969	-0.983	-0.993	-0.979
0.07	-0.993	-0.696	-0.924	-0.877	-0.991	-0.883	-0.991	-0.896	-0.92	-0.978
0.075	-0.993	-0.992	-0.944	-0.937	-0.946	-0.994	-0.835	-0.967	-0.987	-0.945
0.08	-0.877	-0.891	-0.892	-0.947	-0.989	-0.992	-0.927	-0.951	-0.986	-0.992
0.085	-0.807	-0.992	-0.941	-0.902	-0.959	-0.938	-0.962	-0.978	-0.946	-0.952
0.09	-0.993	-0.914	-0.921	-0.92	-0.91	-0.874	-0.799	-0.993	-0.954	-0.904
0.095	-0.987	-0.832	-0.614	-0.899	-0.943	-0.963	-0.992	-0.963	-0.885	-0.931
0.1	-0.945	-0.987	-0.904	-0.939	-0.987	-0.908	-0.933	-0.831	-0.848	-0.831

Рисунок 5 – Случайный поиск для f(x)=(-0.5*cos(0.5x)-0.5)*sin(5x) на [-5;2]

Ссылка на репозиторий с программой – https://github.com/Vumba798/tsisa_lab02

Исходный код программы предоставлен в приложениях 1-4.

Выводы

В результате проделанной работы можно сказать, что метод случайного поиска может использоваться для унимодальных и мультимодальных функций одинаково эффективно. При этом, для увеличения вероятности попадания в заданный интервал или для уменьшения интервала неопределённости требуется увеличить количество точек

Приложение 1. Исходный код файла randomSearcher.hpp

```
#ifndef INCLUDE_RANDOMSEARCHER_HPP
#define INCLUDE_RANDOMSEARCHER_HPP

#include <algorithm>
#include <cmath>
#include <utility>
#include <stdexcept>
#include <vector>

using std::vector;

class RandomSearcher{ protected:
vector<float> _qVec;     vector<float>
_pVec;     vector<vector<uint16_t>>
_nVec;     vector<vector<float>>
_unomodalYvec;     vector<vector<float>>
_multimodalYvec;
   std::pair<float,float> _interval;

    float static _uno_modal_func(const float &x)  {
      return -0.5 * std::cos(0.5 * x) - 0.5;
   }
    float static _multi_modal_func(const float &x) {
return _uno_modal_func(x) * std::sin(5*x);
   }
   inline uint16_t _calculate_N(const float &p, const float &q) const {
return std::ceil(std::log(1-p) / std::log(1-q));
   }
   float _search(const uint16_t& N, float (*f)(const float& x)) const;
template<typename T>     void _print(const vector<vector<T>>
&vec) const; public:
   RandomSearcher(const float &a, const float &b);
void print() const;
};

#endif // INCLUDE_RANDOMSEARCHER_HPP
```

Приложение 2. Исходный код файла randomSearcher.cpp

```
#include <randomSearcher.hpp>
#include <algorithm>
#include <iostream>
#include <iomanip>
#include <utility>
#include <random>
using std::endl; using
std::cout; using
std::cin; using
std::setw; using
std::setfill; using
std::setprecision;
using std::right;

float RandomSearcher::_search(const uint16_t& N, float (*f)(const float& x)) const{
std::vector<float> yVec;     for (size_t i = 0; i < N; ++i) {
       float x = static_cast<float>(std::rand())/static_cast<float>(RAND_MAX);
x = x * (_interval.second - _interval.first) + _interval.first;
yVec.emplace_back(f(x));
   }
   return *std::min_element(yVec.begin(), yVec.end());
}

RandomSearcher::RandomSearcher(const float &a, const float& b) {
if(a >= b){
       throw std::invalid_argument("a must be less than b");
   }
   _interval = std::make_pair(a,b);     for
(float i = 0.005; i < 0.105; i+=0.005){
_qVec.emplace_back(i);
```

```
    }
    for (float i = 0.9; i <= 0.99; i += 0.01){
        _pVec.emplace_back(i);
    }
    for (size_t i = 0; i < _qVec.size(); ++i){
std::vector<uint16_t> tmpNvec;
std::vector<float> tmpUnomodalYvec;
std::vector<float> tmpMultiModalYvec;
for (size_t j = 0; j < _pVec.size(); ++j) {
std::vector<float> multiModalX;
std::vector<float> unoModalX;
            tmpNvec.emplace_back(_calculate_N(_pVec[j], _qVec[i]));
            tmpUnomodalYvec.emplace_back(_search(tmpNvec.back(), _uno_modal_func));
tmpMultiModalYvec.emplace_back(_search(tmpNvec.back(), _multi_modal_func));
        }
        _unomodalYvec.emplace_back(tmpUnomodalYvec);
        _multimodalYvec.emplace_back(tmpMultiModalYvec);
        _nVec.emplace_back(tmpNvec);
    }
}


template <typename T>
void RandomSearcher::_print(const std::vector<std::vector<T>> &vec) const {
std::vector<float> xVec;

    cout << "======================================================="
         << "=======================================================\n";
cout << "|   q\\p   |";
    for(size_t i = 0; i < _pVec.size(); ++i){
cout << " " << setw(7) << _pVec.at(i) << " |";     }

    cout << endl;
```

```
   for(size_t i = 0; i < _qVec.size(); ++i){
cout << "|";
       for(size_t j = 0; j < _pVec.size(); ++j) {
cout << setfill('-') << setw(10) << right << "+";
       }
       cout << setfill(' ') << "---------|" << endl;
cout << "| " << setw(7) << _qVec.at(i) << " |";
for (size_t k = 0; k < _pVec.size(); ++k) {
           cout << " " << setw(7) << setprecision(3) <<  vec.at(i).at(k) << " |";
       }
       cout << endl;
   }
   cout << "-----------------------------------------------------------"
        << "-----------------------------------------------------\n";

}

void RandomSearcher::print() const{
cout << "\t\t\t\t\t\tN PRINT\n";
   _print(_nVec);
   cout << "\n\n\n\t\t\t\t\t\tUNOMODAL PRINT\n";
   _print(_unomodalYvec);
   cout << "\n\n\n\t\t\t\t\t\tMULTIMODAL PRINT\n";
   _print(_multimodalYvec);
}
```

Приложение 3. Исходный код файла main.cpp

```
#include <iostream>
#include <string>
#include <randomSearcher.hpp>

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

int main(int argc, char**
argv){     RandomSearcher rs (-
5, 2);     rs.print();     return 0;
}
```

Приложение 4. Исходный код файла CMakeLists.txt

```
cmake_minimum_required(VERSION 3.4) set(CMAKE_CXX_STANDARD
11) set(CMAKE_CXX_STANDARD_REQUIRED ON)

project(RandomSearcher)

add_library(random_searcher STATIC
   ${CMAKE_CURRENT_SOURCE_DIR}/sources/randomSearcher.cpp
)
add_executable(main
 ${CMAKE_CURRENT_SOURCE_DIR}/sources/main.cpp
)

target_include_directories(random_searcher
   PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}/include
)

target_include_directories(main
   PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}/include
)

target_link_libraries(main PUBLIC random_searcher)
```

.

Контрольные вопросы

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

Разделяют направленный и ненаправленный метод случайного поиска.

В случае направленного поиска, последующие измерения напрямую зависят от предыдущих и случайность здесь заключается лишь в задании направления спуска. Поэтому, как правило, в условиях мультимодальных функций данный поиск приводит лишь к локальным экстремумам, хотя сходимость данного метода выше, чем у ненаправленного случайного поиска.

Ненаправленный случайный поиск заключается в случайном выборе N точек и последующем вычислении минимального среди них значения. Этот метод показывает низкую эффективность в случае анализа уномодальных функций, однако главное преимущество данного способа заключается в поиске глобального экстремума мудьтимодальных функций.