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 № 1
по дисциплине «Теория систем и системный анализ»

Тема: «Исследование методов прямого поиска экстремума унимодальной функции одного переменного»

Вариант 1

Выполнил: Александров А.А.,
студент группы ИУ8-31

Проверил: Коннова Н.С.,
доцент каф. ИУ8

Москва 2020

1. Цель работы

Исследовать функционирование и провести сравнительный анализ различных алгоритмов прямого поиска экстремума (пассивный поиск, метод дихотомии, золотого сечения, Фибоначчи) на примере унимодальной функции одного переменного.

2. Условие задачи

На интервале [-5; 2] задана унимодальная функция одного переменного

$$f(x) = -0.5\cos(0.5x) - 0.5$$

Используя метод дихотомии, найти интервал нахождения минимума $f(x)$ с заданным количеством итераций. Провести сравнение с методом оптимального пассивного поиска. Результат, в зависимости от числа точек разбиения N, представить в виде таблицы.

3. Ход работы

Построим график заданной функции и определим местонахождение её минимума:

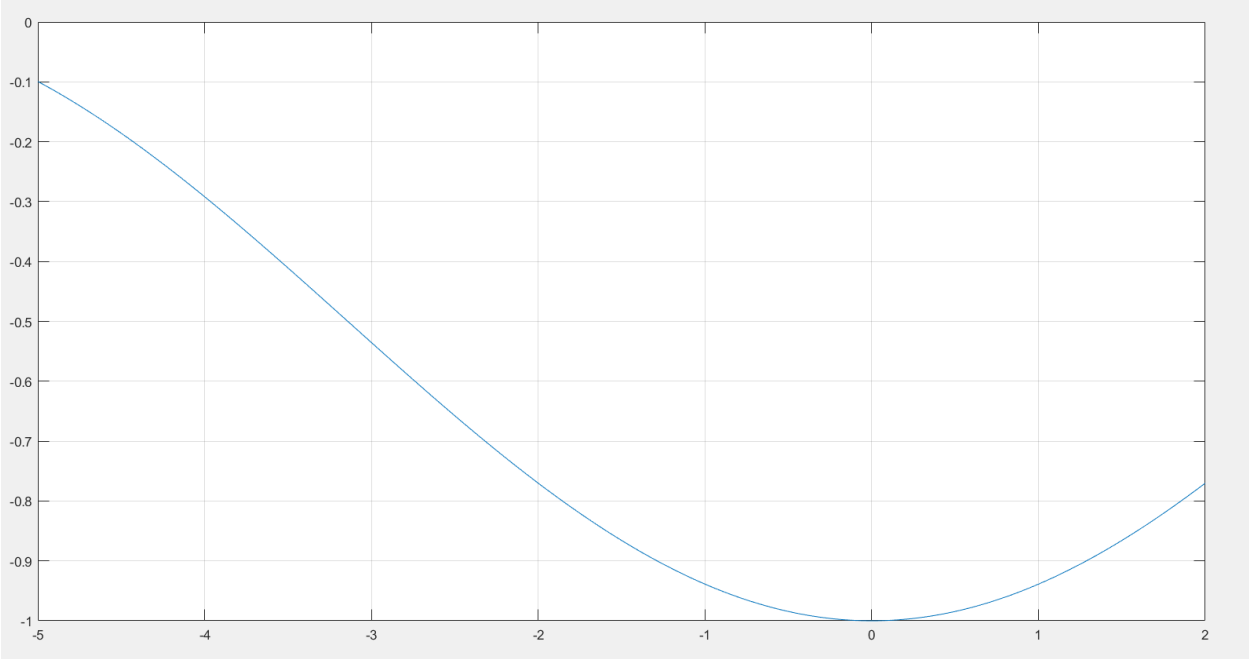

Рисунок 1 - График функции $f(x) = -0.5\cos(0.5x) - 0.5$ на интервале [-5, 2]

Как видно из графика, функция достигает своего минимума в точке $x = 0$. Теперь проведём программный расчет при помощи методов оптимального пассивного поиска и дихотомии.

Результат работы программы представлен в таблицах 1 и 2:

Таблица 1 – результат работы метода пассивного поиска

```
PASSIVE SEARCHER

============================================================
```

Amount of points	Minimal value	Uncertainty interval
1	-0.865844	7
2	-0.993072	4.66667
3	-0.996099	3.5
4	-0.977668	2.8
5	-0.993072	2.33333
6	-1	2
7	-0.996099	1.75
8	-0.993072	1.55556
9	-0.999375	1.4
10	-0.999484	1.27273
11	-0.996099	1.16667
12	-0.998521	1.07692
13	-1	1
14	-0.998889	0.933333
15	-0.997804	0.875
16	-0.999784	0.823529
17	-0.999807	0.777778
18	-0.998443	0.736842
19	-0.999375	0.7
20	-1	0.666667
21	-0.999484	0.636364
22	-0.998937	0.608696
23	-0.999892	0.583333
24	-0.9999	0.56
25	-0.999168	0.538462
26	-0.999657	0.518519

27	-1	0.5
28	-0.999703	0.482759
29	-0.999375	0.466667
30	-0.999935	0.451613
31	-0.999939	0.4375
32	-0.999484	0.424242
33	-0.999784	0.411765
34	-1	0.4
35	-0.999807	0.388889
36	-0.999589	0.378378
37	-0.999957	0.368421
38	-0.999959	0.358974
39	-0.999648	0.35
40	-0.999851	0.341463
41	-1	0.333333
42	-0.999865	0.325581
43	-0.999709	0.318182
44	-0.999969	0.311111
45	-0.99997	0.304348
46	-0.999745	0.297872
47	-0.999892	0.291667
48	-1	0.285714
49	-0.9999	0.28
50	-0.999784	0.27451
51	-0.999977	0.269231
52	-0.999978	0.264151
53	-0.999807	0.259259
54	-0.999917	0.254545

55	-1	0.25
56	-0.999923	0.245614
57	-0.999833	0.241379
58	-0.999982	0.237288
59	-0.999983	0.233333
60	-0.999849	0.229508
61	-0.999935	0.225806
62	-1	0.222222
63	-0.999939	0.21875
64	-0.999867	0.215385
65	-0.999986	0.212121
66	-0.999986	0.208955
67	-0.999878	0.205882
68	-0.999947	0.202899
69	-1	0.2
70	-0.99995	0.197183
71	-0.999892	0.194444
72	-0.999988	0.191781
73	-0.999989	0.189189
74	-0.9999	0.186667
75	-0.999957	0.184211
76	-1	0.181818
77	-0.999959	0.179487
78	-0.99991	0.177215
79	-0.99999	0.175
80	-0.99999	0.17284
81	-0.999916	0.170732
82	-0.999964	0.168675

83	-1	0.166667
84	-0.999965	0.164706
85	-0.999924	0.162791
86	-0.999992	0.16092
87	-0.999992	0.159091
88	-0.999929	0.157303
89	-0.999969	0.155556
90	-1	0.153846
91	-0.99997	0.152174
92	-0.999935	0.150538
93	-0.999993	0.148936
94	-0.999993	0.147368
95	-0.999939	0.145833
96	-0.999973	0.14433
97	-1	0.142857
98	-0.999974	0.141414
99	-0.999944	0.14
100	-0.999994	0.138614
101	-0.999994	0.137255
102	-0.999947	0.135922
103	-0.999977	0.134615
104	-1	0.133333
105	-0.999978	0.132075
106	-0.999951	0.130841
107	-0.999995	0.12963
108	-0.999995	0.12844
109	-0.999954	0.127273
110	-0.99998	0.126126

111	-1	0.125
112	-0.99998	0.123894
113	-0.999957	0.122807
114	-0.999995	0.121739
115	-0.999995	0.12069
116	-0.999959	0.119658
117	-0.999982	0.118644
118	-1	0.117647
119	-0.999983	0.116667
120	-0.999962	0.115702
121	-0.999996	0.114754
122	-0.999996	0.113821
123	-0.999963	0.112903
124	-0.999984	0.112
125	-1	0.111111
126	-0.999985	0.110236
127	-0.999966	0.109375
128	-0.999996	0.108527
129	-0.999996	0.107692
130	-0.999967	0.10687
131	-0.999986	0.106061
132	-1	0.105263
133	-0.999986	0.104478
134	-0.999969	0.103704
135	-0.999997	0.102941
136	-0.999997	0.10219
137	-0.99997	0.101449
138	-0.999987	0.100719

139	-1	0.1

Таблица 2 – результат работы метода дихотомии

DICHOTOMY SEARCHER

Left bound	Right bound	x1	x2	f(x1)	f(x2)	length	step Number
-5	2	-1.533	-1.467	-0.86011	-0.87147	7	1
-1.53333	2	0.2	0.2667	-0.9975	-0.99556	3.5333	2
-1.53333	0.266667	-0.6667	-0.6	-0.97248	-0.97767	1.8	3
-0.666667	0.266667	-0.2333	-0.1667	-0.9966	-0.99826	0.93333	4
-0.233333	0.266667	-0.01667	0.05	-0.99998	-0.99984	0.5	5
-0.233333	0.05	-0.125	-0.05833	-0.99902	-0.99979	0.28333	6
-0.125	0.05	-0.07083	-0.004167	-0.99969	-1	0.175	7
-0.0708333	0.05	-0.04375	0.02292	-0.99988	-0.99997	0.12083	8
-0.04375	0.05	-0.03021	0.03646	-0.99994	-0.99992	0.09375	9

Построим графики зависимостей интервала неопределённости от числа точек N (для оптимального пассивного поиска и для метода дихотомии).

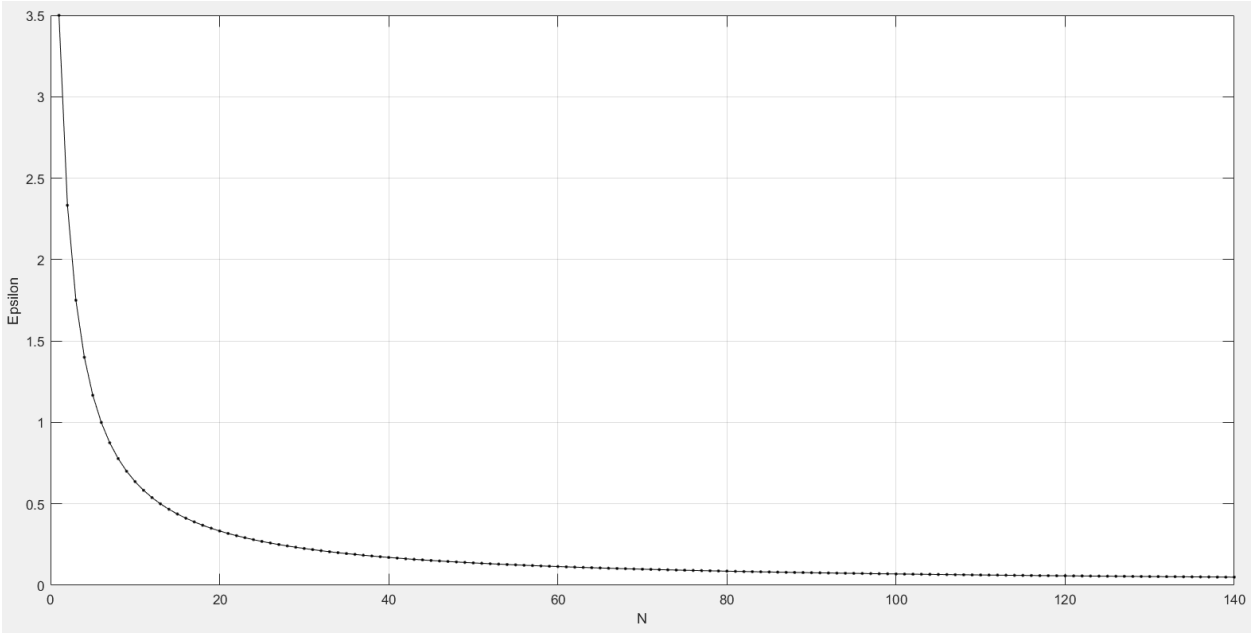

Рисунок 2 – зависимость интервала неопределённости от кол-ва измерений, метод оптимально пассивного поиска

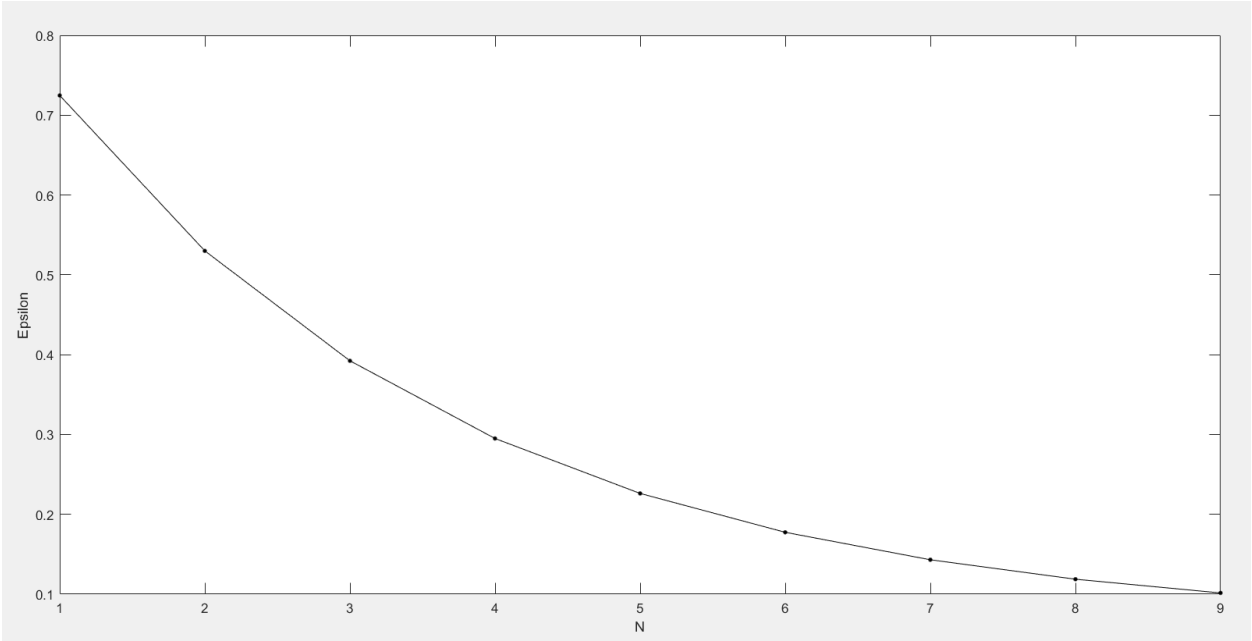

Рисунок 3 - зависимость интервала неопределённости от кол-ва измерений, метод дихотомии

Ссылка на git-репозиторий: https://github.com/Vumba798/tsisa_lab01
Исходный код программы приведён в приложениях 1 - 6.

4. Выводы

В конечном итоге расчёт разными способами показал, что метод дихотомии значительно эффективнее метода оптимально пассивного поиска при нахождении экстремума унимодальной функции одного переменного.

Приложение 1. Исходный код файла searcher.hpp

```
#ifndef INCLUDE_SEARCHER_HPP
#define INCLUDE_SEARCHER_HPP

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



class Searcher{
protected:
   float _epsilon = 0.1;
   std::pair<float,float> _interval;

   inline float _func(const float &x) const noexcept{
      return -0.5 * std::cos(0.5 * x) - 0.5;
   }

public:
   inline Searcher(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);
   };
   virtual void print() const = 0;
};

#endif // INCLUDE_SEARCHER_HPP
```

Приложение 2. Исходный код файла passiveSearcher.hpp

```
#ifndef INCLUDE_PASSIVESEARCHER_HPP_
#define INCLUDE_PASSIVESEARHCER_HPP_
#include <searcher.hpp>
#include <vector>

class PassiveSearcher : public Searcher{
public:
   inline PassiveSearcher(float a, float b) : Searcher(a, b) {};
   std::vector<std::pair<float, float>> search() const;
   void print() const override final;
};

#endif // INCLUDE_PASSIVESEARHCER_HPP_
```

Приложение 3. Исходный код файла dichotomySearcher.hpp

```
#ifndef INCLUDE_DICHOTOMYSEARCHER_HPP_
#define INCLUDE_DICHOTOMYSEARCHER_HPP_

#include "searcher.hpp"
#include <vector>

struct Result{
   float intervalFirst;
   float intervalSecond;
   float X1;
   float X2;
   float functionX1;
   float functionX2;
   float length;
   size_t stepNum;

   inline Result(
         float interFirst, float interSecond,
         float x1, float x2,
         float funcX1, float funcX2,
         float len, size_t num) :
      intervalFirst(interFirst), intervalSecond(interSecond),
      X1(x1), X2(x2),
      functionX1(funcX1), functionX2(funcX2),
      length(len), stepNum(num) {};
};

class DichotomySearcher : public Searcher{
private:
```

```
    float _delta;
public:
    inline DichotomySearcher(const int &a, const int &b) :
        Searcher(a, b),
        _delta(_epsilon / 3.0) {};

    std::vector<Result> search() const;
    void print() const override final;

};

#endif // INCLUDE_DICHOTOMYSEARCHER_HPP_
```

Приложение 4. Исходный код файла passiveSearcher.cpp

```
#include <passiveSearcher.hpp>
#include <algorithm>
#include <iomanip>
#include <iostream>



using std::endl;
using std::cout;
using std::setw;
using std::setfill;

std::vector<std::pair<float, float>> PassiveSearcher::search() const{
   float l; // l - interval of uncertainty
   size_t N = 1; // N - amount of points

   // first is a value, second is an interval of uncertainty
   std::vector<std::pair<float, float>> minElementVec;

   do{
      l = 2.0 / (N+1) * (_interval.second-_interval.first);
      std::vector<float> tmpVec;
      for (size_t k = 1; k <= N; ++k){
         float x = static_cast<float>(k)/(N+1) * (_interval.second-_interval.first) + _interval.first;
         tmpVec.emplace_back(_func(x));
      }
      float min = *std::min_element(tmpVec.begin(), tmpVec.end());
      minElementVec.emplace_back(std::make_pair(min, l));

      ++N;
```

```
    } while (l > _epsilon);

    return minElementVec;
}

void PassiveSearcher::print() const{
    auto vec = search();
    cout << "| Amount of points | " << "Minimal value" << " | " <<
        setw(6) << "Uncertainty interval |\n";
    cout << "|" << setfill('-') << setw(57) << "" <<  "|" << endl;
    for (size_t i = 0; i < vec.size(); ++i){
        cout << setfill(' ') << setw(0);
        cout << "| " << setw(16) << i + 1 << " | " <<
            setw(13) <<  vec[i].first << " | " <<
            setw(20) << vec[i].second << " |\n";
    }
    cout << "|" << setfill('-') << setw(57) << "" <<  "|" << endl;
}
```

Приложение 5. Исходный код файла dichotomySearcher.cpp

```
#include <dichotomySearcher.hpp>
#include <utility>
#include <iomanip>
#include <iostream>

using std::endl;
using std::cin;
using std::cout;
using std::setfill;
using std::setw;
using std::right;
using std::setprecision;

std::vector<Result> DichotomySearcher::search() const{
   float l;
   float intervalFirst = _interval.first;
   float intervalSecond = _interval.second;

   std::vector<Result> resultVec;
   size_t stepNum = 1;
   do{
      l = intervalSecond - intervalFirst;
      float x1 = intervalFirst + l/2 - _delta;
      float x2 = intervalFirst + l/2 + _delta;
      float fx1 = _func(x1);
      float fx2 = _func(x2);

      resultVec.emplace_back(Result(
            intervalFirst, intervalSecond,
```

```
                x1, x2,
                fx1, fx2,
                l, stepNum));

        if (fx1 < fx2){
            intervalSecond = x2;
        }else{
            intervalFirst = x1;
        }

        ++stepNum;

    } while (l > _epsilon);

    return resultVec;
}

void DichotomySearcher::print() const{
    auto resultVec = search();

    cout << "| Left bound | Right bound |"
        <<"    x1    |    x2    |    f(x1)    |" <<
        "    f(x2)    |  length  | step Number |\n";
    cout << "|" << setfill('-') << setw(107) << right << "|\n";
    for (size_t i = 0; i < resultVec.size(); ++i){
        cout << setfill(' ');
        cout << "| " << setw(10) << setprecision(6) << resultVec[i].intervalFirst << " | " <<
            setw(11) << setprecision(6) << resultVec[i].intervalSecond << " | " <<
            setw(10) << setprecision(4) << resultVec[i].X1 << " | " <<
            setw(10) << resultVec[i].X2 << " | " <<
            setw(11) << setprecision(5) << resultVec[i].functionX1 << " | " <<
            setw(11) << resultVec[i].functionX2 << " | " <<
            setw(8) << setprecision(5) << resultVec[i].length << " | " <<
```

```
        setw(11) << resultVec[i].stepNum << " |\n";
    cout << "|" << setfill('-') << setw(107) << right << "|\n";
    }
}
```

Приложение 6. Исходный код файла main.cpp

```
#include <iostream>
#include <string>
#include <passiveSearcher.hpp>
#include <dichotomySearcher.hpp>

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

int main(int argc, char** argv){
    cout << "The function is: func(x) = -0.5*cos(0.5x) - 0.5" << endl << endl;
    cout << endl << "\t\t\tPASSIVE SEARCHER" << endl;
    cout << "============================================================" << endl;
    PassiveSearcher ps(-5, 2);
    ps.print();

    cout << endl << endl << endl << "\t\t\t\t\tDICHOTOMY SEARCHER" << endl;
    cout << "==========================================================";
    cout << "==================================================" << endl;
    DichotomySearcher ds(-5, 2);
    ds.print();
    return 0;
}
```

Приложение 6. Исходный код файла CMakeLists.txt

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

project(Searcher)

add_library(searcher STATIC
   ${CMAKE_CURRENT_SOURCE_DIR}/sources/passiveSearcher.cpp
   ${CMAKE_CURRENT_SOURCE_DIR}/sources/dichotomySearcher.cpp
)
add_executable(main
  ${CMAKE_CURRENT_SOURCE_DIR}/sources/main.cpp
)

target_include_directories(searcher
   PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}/include
)

target_include_directories(main
   PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}/include
)

target_link_libraries(main PUBLIC searcher)
```