МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Московский государственный технический университет

им. Н.Э. Баумана

(МГТУ им. Н.Э. Баумана)

Кафедра «Информационная безопасность» (ИУ8)

Лабораторная работа № 1

Исследование методов прямого поиска экстремума унимодальной функции одного переменного

Выполнил: Броцкий К.А.,

студент группы ИУ8-32

Проверил: Коннова Н.С.,

доцент каф. ИУ8

г. Москва 2020 г.

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.

Постановка задачи

На интервале [a,b] задана унимодальная функция одного переменного f (x) . Используя методы последовательного поиска (дихотомии, золотого сечения и Фибоначчи), найти интервал нахождения минимума f(x) при заданной наибольшей допустимой длине интервала неопределенности ε = 0,1. Провести сравнение с методом оптимального пассивного поиска (п. 2.2 в [1]). Результат, в зависимости от числа точек разбиения N, представить в виде таблицы.

Вариант 1:

№пп	Функция $f(x)$	a	b	Метод поиска	
1	$-0,5\cos 0,5x - 0,5$	–5	2	опт. пассивный	дихотомия

График функции

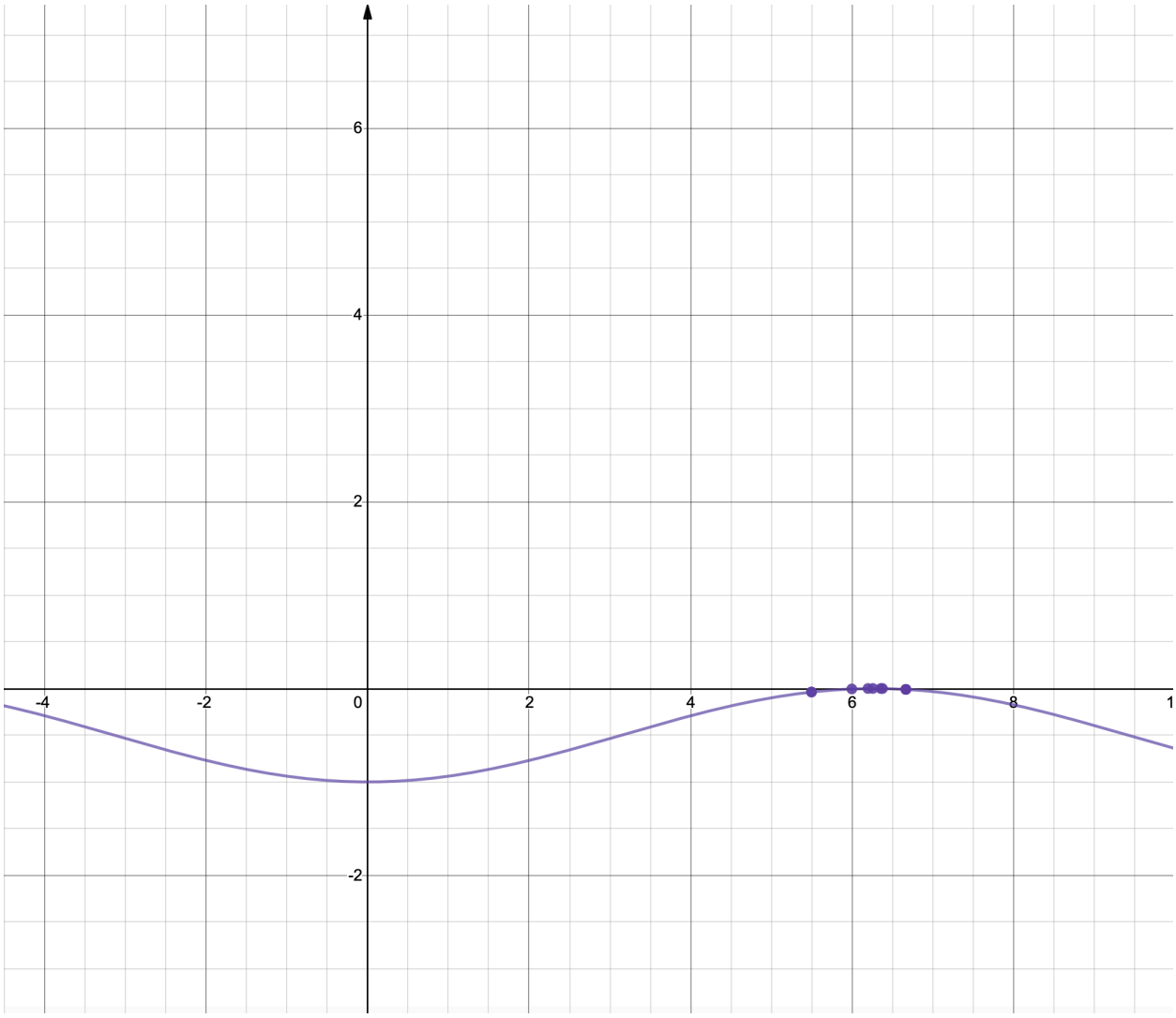

Рис.1. График функции

Скриншот консоли

```
Variant 1.
Function: -0,5*cos0,5x-0,5
Interval: [-5 2]

Part 1. Finding minimum of the function with dichotomy method
```

Start of the interval (ak)	End of the interval (bk)	Length of the interval (l)	f(ak)	f(bk)
-5	2	7	-0.0994282	-0.770151
-1.49	2	3.49	-0.867544	-0.770151
-1.49	0.245	1.735	-0.867544	-0.996253
-0.6125	0.245	0.8575	-0.976735	-0.996253
-0.17375	0.245	0.41875	-0.998114	-0.996253
-0.17375	0.025625	0.199375	-0.998114	-0.999959
-0.0640625	0.025625	0.0896875	l < epsilon	

```
Minimum is reached at the point x = -0.0192 +- 0.045

Part 2. Finding minimum of the function with optimal passive finding method
```

Number of points (N)	Value of x in the minimum
1	5.5 +- 3.5
2	6.67 +- 2.33
3	5.5 +- 1.75
4	6.2 +- 1.4
5	6.67 +- 1.17
6	6 +- 1
7	6.38 +- 0.875
8	6.67 +- 0.778
9	6.2 +- 0.7
10	6.45 +- 0.636
11	6.08 +- 0.583
12	6.31 +- 0.538
13	6.5 +- 0.5
14	6.2 +- 0.467
15	6.38 +- 0.438
16	6.12 +- 0.412
17	6.28 +- 0.389
18	6.42 +- 0.368
19	6.2 +- 0.35
20	6.33 +- 0.333
21	6.14 +- 0.318
22	6.26 +- 0.304
23	6.38 +- 0.292
24	6.2 +- 0.28
25	6.31 +- 0.269
26	6.41 +- 0.259
27	6.25 +- 0.25
28	6.34 +- 0.241
29	6.2 +- 0.233
30	6.29 +- 0.226
31	6.38 +- 0.219
32	6.24 +- 0.212
33	6.32 +- 0.206
34	6.2 +- 0.2
35	6.28 +- 0.194
36	6.35 +- 0.189
37	6.24 +- 0.184
38	6.31 +- 0.179
39	6.2 +- 0.175
40	6.27 +- 0.171
41	6.33 +- 0.167
42	6.23 +- 0.163
43	6.3 +- 0.159
44	6.36 +- 0.156
45	6.26 +- 0.152
46	6.32 +- 0.149
47	6.23 +- 0.146
48	6.29 +- 0.143
49	6.34 +- 0.14
50	6.25 +- 0.137
51	6.31 +- 0.135
52	6.23 +- 0.132
53	6.28 +- 0.13
54	6.33 +- 0.127
55	6.25 +- 0.125
56	6.3 +- 0.123
57	6.22 +- 0.121
58	6.27 +- 0.119
59	6.32 +- 0.117
60	6.25 +- 0.115
61	6.29 +- 0.113
62	6.33 +- 0.111
63	6.27 +- 0.109
64	6.31 +- 0.108
65	6.24 +- 0.106
66	6.28 +- 0.104
67	6.32 +- 0.103
68	6.26 +- 0.101
69	6.3 +- 0.1

Рис.2. Скриншот консоли

Листинг программы с реализацией алгоритмов на C++

```
#include <cmath>
#include <iomanip>
#include <iostream>
#include <sstream>

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

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

void beautifulPrintingForPart1(const double ak, const double bk) {
    cout << '|' << std::setw(13) << ak << ' '
        << '|' << std::setw(13) << bk << ' '
        << '|' << std::setw(13) << bk - ak << ' '
        << '|' << std::setw(13) << myFunctionFromTask(ak) << ' '
        << '|' << std::setw(13) << myFunctionFromTask(bk) << ' ' << '|' << '\n';
}

void dichotomy(double lower, double upper,
        const double epsilon, const double delta) {
    cout << "\nPart 1. Finding minimum of the function with dichotomy
method\n"
         << std::string(76, '_') << '\n'
         << '|' << std::string(3, ' ') << "Start of" << std::string(3, ' ')
         << '|' << std::string(4, ' ') << "End of" << std::string(4, ' ')
         << '|' << std::string(2, ' ') << "Length of" << std::string(3, ' ')
         << '|' << std::string(14, ' ')
         << '|' << std::string(14, ' ') << '|' << '\n'
         << '|' << std::string(1, ' ') << "the interval" << std::string(1, ' ')
         << '|' << std::string(1, ' ') << "the interval" << std::string(1, ' ')
         << '|' << std::string(1, ' ') << "the interval" << std::string(1, ' ')
         << '|' << std::string(4, ' ') << "f(ak)" << std::string(5, ' ')
         << '|' << std::string(4, ' ') << "f(bk)" << std::string(5, ' ') << '|' << '\n'
         << '|' << std::string(5, ' ') << "(ak)" << std::string(5, ' ')
         << '|' << std::string(5, ' ') << "(bk)" << std::string(5, ' ')
         << '|' << std::string(5, ' ') << "(l)" << std::string(6, ' ')
         << '|' << std::string(14, ' ')
         << '|' << std::string(14, ' ') << '|' << '\n'
         << std::string(76, '-') << '\n';
```

```
    while (upper - lower > epsilon) {
        beautifulPrintingForPart1(lower, upper);
        double x1 = lower + (upper - lower) / 2 - delta,
               x2 = lower + (upper - lower) / 2 + delta;
        myFunctionFromTask(x1) < myFunctionFromTask(x2)
        ? upper = x1
        : lower = x2;
    }
    cout << '|' << std::setw(13) << lower << ' '
         << '|' << std::setw(13) << upper << ' '
         << '|' << std::setw(13) << upper - lower << ' '
         << '|' << std::string(9, ' ') << "I < epsilon" << std::string(9, ' ') << '|' <<
'\n'
         << std::string(76, '-') << '\n'
         << "Minimum is reached at the point x = " << std::setprecision(3)
           << lower + (upper - lower) / 2 << " +- " << std::setprecision(2)
           << (upper - lower) / 2 << '\n';
}

void optimalPassiveFinding(const double lower, const double upper,
                           const double epsilon) {
    cout << "\nPart 2. Finding minimum of the function with optimal passive
finding method\n"
         << std::string(27, '_') << '\n'
         << '|' << "Number of " << '|' << "  Value of x  " << '|' << '\n'
         << '|' << "points (N)" << '|' << "in the minimum" << '|' << '\n'
         << std::string(27, '-') << '\n';

    size_t N = 1;
    double finding;
    while ((upper - lower) / N > epsilon) {
        double x = upper;
        finding = x;
        for (size_t i = 0; i < N; ++i) {
            x += (upper - lower) / (N + 1);
            if (myFunctionFromTask(x) > myFunctionFromTask(finding))
                finding = x;
        }

        std::ostringstream os;
        os << std::setw(5) << std::setprecision(3) << finding << " +- "
           << std::setprecision(3) << (upper - lower) / (N + 1);

        cout << '|' << std::setw(6) << N << std::setw(4) << " |"
           << std::left << std::setw(15) << os.str() << "|\n" << std::right;
```

```
        ++N;
    }
    cout << std::string(27, '-') << '\n';
}

const double LOWER_EDGE = -5.;
const double UPPER_EDGE = 2.;
const double EPSILON = .1;

int main() {
    cout << "Variant 1.\nFunction: -0,5*cos0,5x-0,5\nInterval: [" <<
LOWER_EDGE << " " << UPPER_EDGE << "]\n";

    dichotomy(LOWER_EDGE, UPPER_EDGE, EPSILON, .01);  // Part 1.
Dichotomy
    optimalPassiveFinding(LOWER_EDGE, UPPER_EDGE, EPSILON);  //
Part 2. Optimal passive finding


    return 0;
}
```

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.