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 4

Выполнил: Бояркина Е. Р.,
студент группы ИУ8-31

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

г. Москва,
2020 г.

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 $[-2;\ 0]$ задана унимодальная функция одного переменного $f(x) = cos(x)\,tanh(x)$. Используя метод дихотомии, найти интервал нахождения минимума $f(x)$ при заданной наибольшей допустимой длине интервала неопределенности $\varepsilon = 0{,}1$. 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*):

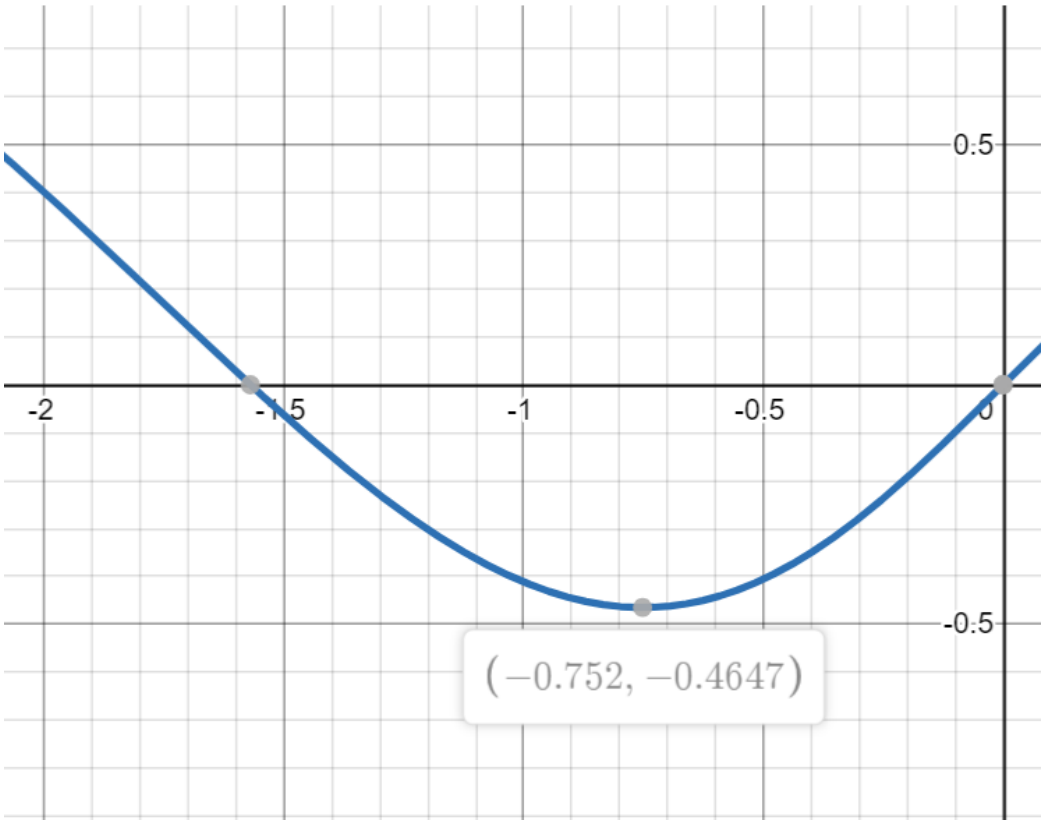

Рисунок 1 – График функции *f(x) = cos(x)tanh(x)* на интервале [-2; 0]

Как видно из графика, функция достигает своего минимума в точке $x = -0{,}752$. Теперь проведём программный расчет при помощи методов дихотомии и оптимального пассивного поиска.

Реализация метода дихотомии:

Отрезок поиска делится пополам точкой x. Вычисляются значения функции на границах окрестности точки x: $(x - 0.01, x + 0.01)$. Исключается левая половина, если значение функции в точке левой границы окрестности больше, чем в правой. Иначе, исключается правая половина. Эти действия повторяются до тех пор, пока отрезок поиска будет больше отрезка неопределенности.

Реализация оптимального пассивного поиска:

Количество точек N на 1 меньше, чем количество отрезков.

$N = \frac{(b-a)}{\varepsilon} - 1$, где ε – наибольшая длина интервала неопределенности

Для интервала неопределенности 0.1 погрешность равна 0.05. Количество точек равно 19.

Точки расположены равномерно по отрезку, следовательно, координата точки с номером k:

$$x_k = \frac{k}{N+1}(b - a)$$

Результат работы программы представлен в таблицах 1 и 2:

Таблица 1 – результат работы дихотомии

Start of the interval (ak)	End of the interval (bk)	Length of the interval (l)	f(ak)	f(bk)
-2	0	2	0.401177	0
-0.99	0	0.99	-0.415557	0
-0.99	-0.505	0.485	-0.415557	-0.407867
-0.99	-0.7575	0.2325	-0.415557	-0.464707
-0.86375	-0.7575	0.10625	-0.453534	-0.464707
-0.800625	-0.7575	0.043125	l < epsilon	

Минимум достигается в точке x = -0.779 +- 0.022

Таблица 2 – результат работы оптимального пассивного поиска

Number of points (N)	Value of x in the minimum
1	1 +- 1
2	0.667 +- 0.667
3	1 +- 0.5
4	0.8 +- 0.4
5	0.667 +- 0.333
6	0.857 +- 0.286
7	0.75 +- 0.25
8	0.667 +- 0.222
9	0.8 +- 0.2
10	0.727 +- 0.182
11	0.833 +- 0.167
12	0.769 +- 0.154
13	0.714 +- 0.143
14	0.8 +- 0.133
15	0.75 +- 0.125
16	0.706 +- 0.118
17	0.778 +- 0.111
18	0.737 +- 0.105
19	0.8 +- 0.1

Построим график зависимостей погрешности от числа точек N (для дихотомии – *Рисунок 2*, для оптимального пассивного поиска – *Рисунок 3*):

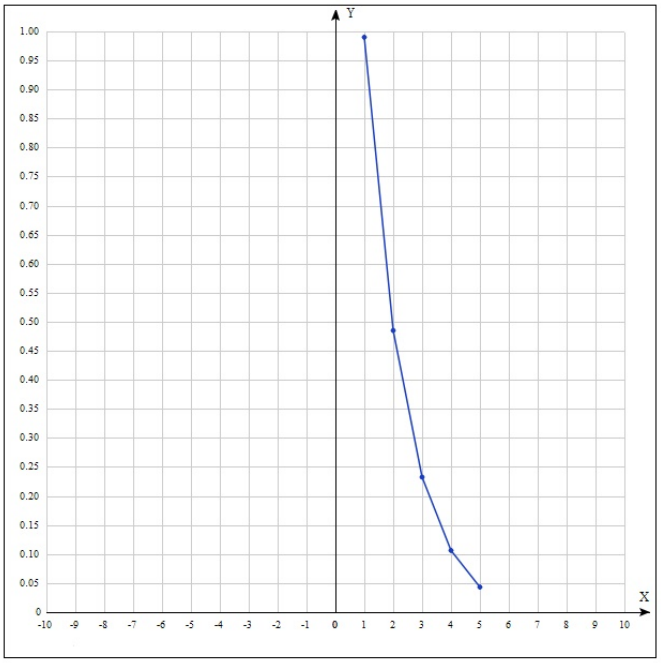

Рисунок 2 – График зависимости погрешности от числа точек N для дихотомии

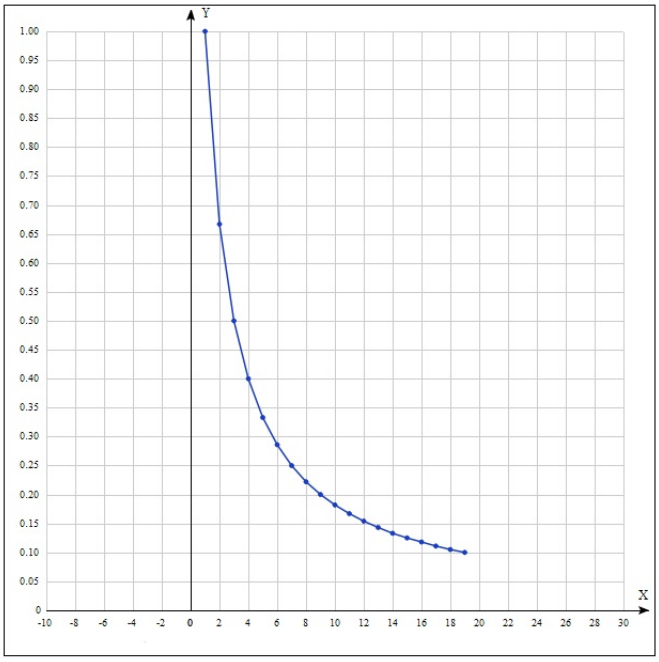

Рисунок 3 – График зависимости погрешности от числа точек N для оптимального пассивного поиска

Ссылка на git-репозиторий: https://github.com/freesummerwind/tsisa_lab_01

4. Выводы

Из полученных таблиц и графиков видно, что метод дихотомии значительно эффективнее метода пассивного поиска при поиске экстремума унимодальной функции одного переменного.

Приложение 1. Исходный код программы

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

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

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

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, ' ') << "l < 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 = -2.;
const double UPPER_EDGE = 0.;
const double EPSILON = .1;

int main() {
    cout << "Variant 4.\nFunction: cos(x)*th(x)\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;
}
```